

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

УТВЕРЖДАЮ

Зав.кафедрой,

к. ф.-м. н., доцент

_____ С. В. Миронов

ОТЧЕТ О ПРАКТИКЕ

студента 2 курса 273 группы факультета КНиИТ

Козырева Юрия Дмитриевича

вид практики: учебная

кафедра: математической кибернетики и компьютерных наук

курс: 2

семестр: 2

продолжительность: 2 нед., с _____ г. по _____ г.

Руководитель практики от университета,

зав. каф., к. ф.-м. н., доцент _____

С. В. Миронов

Руководитель практики от организации (учреждения, предприятия),

доцент, к. ф.-м. н. _____

С. В. Миронов

Тема практики: «МОДЕЛИРОВАНИЕ ДИНАМИКИ ХАРАКТЕРИСТИК
УЗЛОВ В СЛОЖНЫХ СЕТЯХ»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Теоретические сведения	6
1.1 Случайные графы	6
1.2 Индекс дружбы	6
1.3 Масштабируемые метрики	8
1.3.1 Химическое расстояние	9
1.3.2 DeepSIM	9
1.3.3 Визуальный анализ	9
1.3.4 ANNR	10
2 Модификации индекса дружбы	15
3 Экспериментальная проверка гипотезы	17
4 Анализ результатов	19
ЗАКЛЮЧЕНИЕ	21
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	22
Приложение А Текст программы для проведения эксперимента по сравнению ANND и ANNR	24
Приложение Б Текст программы для проведения эксперимента по сравнению индекса дружбы и его модификаций	30

ВВЕДЕНИЕ

В повседневной жизни для решения многих задач часто используются случайные графы. Случайные графы нашли практическое применение во всех областях, где нужно смоделировать сложные сети. Имеется большое число моделей случайных графов, отражающих разнообразные типы сложных сетей в различных областях. Случайные графы применяются при моделировании и анализе биологических и социальных систем, сетей, а также при решении многих задач класса NP.

Случайные графы впервые определены венгерскими математиками П. Эрдёшем и А. Реньи в книге 1959 года «On Random Graphs» [1] и независимо американским математиком, Э. Гильбертом, в его статье «Random graphs» [2].

Случайный граф — общий термин для обозначения вероятностного распределения графов [3]. Их можно описать просто распределением вероятности или случайным процессом, создающим эти графы.

Теория случайных графов находится на стыке комбинаторики, теории графов и теории вероятностей. В основе ее лежит глубокая идея о том, что мощные инструменты современной теории вероятностей должны поспособствовать более верному осознанию природы графа, призваны помочь решению многих комбинаторных и теоретико-графовых задач [4].

С математической точки зрения случайные графы необходимы для ответа на вопрос о свойствах типичных графов. Для этого используются модели Эрдёша—Реньи, Барабаши—Альберт, модель триадного замыкания, модель Бьянкони-Барабаши и другие. Все модели основываются на различных свойствах социальных сетей.

При практическом применении случайных графов для моделирования систем, содержащих миллионы вершин необходима возможность определять пригодность той или иной модели для рассматриваемой симуляции, не прибегая к необходимости построения модели в полную величину. Для этого необходима локальная метрика позволяющая сравнивать графы разных размеров.

Цель настоящей работы — предложить потенциальный вариант локальной масштабируемой метрики графа. Для достижения этой цели необходимо решить следующие задачи:

- анализ существующих масштабируемых метрик графов;
- формулировка локальной масштабируемой метрики графа;

— реализация и анализ предложенной метрики.

1 Теоретические сведения

1.1 Случайные графы

Случайные графы широко применяются во многих сферах человеческой деятельности.

Некоторые протоколы передачи данных в компьютерных сетях основаны на различных свойствах случайных графов и в данный момент активно ведутся исследования по усовершенствованию этих алгоритмов. Например, Алессандро Больоло и Кристель Сирокки в своей статье «Topological network features determine convergence rate of distributed average algorithms» [5] изучают топологию и различные метрики графов в компьютерных сетях и Gossip-алгоритмах для разработки более эффективной замены для Gossip-протоколов. Gossip — это группа протоколов в одноранговой компьютерной коммуникации, в которых распространение информации идёт способом, схожим с образом распространения эпидемий, и сводящимся к тому, что каждый или некоторые из узлов могут передавать обновляемые данные известным этому узлу соседям. Основными преимуществами данных алгоритмов являются их надёжность и отказоустойчивость, однако передача большого количества избыточных данных приводит к с лишком высокой нагрузке на оборудование.

Существует множество различных моделей построения случайных графов [6–11]. И для определения какую модель лучше использовать при моделировании той или иной системы, необходима метрика позволяющая сравнивать графы.

1.2 Индекс дружбы

Ранее одной из основных метрик используемых для сравнения графов являлся индекс дружбы. Понятие индекса дружбы тесно связано с парадоксом дружбы.

С момента появления социальных сетей — Facebook, Vkontakte, LiveJournal, Instagram, LinkedIn, MySpace и т. д. прошло не так много времени, но они уже плотно вошли в повседневную жизнь многих людей.

Опросы показывают, что 76% пользователей Интернета в России (по данным агентства PRT на январь 2014) и примерно 73% жителей Соединенных Штатов являются активными пользователями социальных сетей, и эта цифра растёт [12].

В современном обществе социальные сети становятся огромной базой информации, которую ученые и работодатели все чаще привлекают для решения конкретных задач, будь то научное исследование или оценка кандидата на определенную должность.

Научный интерес к изучению пользователей социальных сетей стремительно растет. На данный момент накоплено большое количество эмпирического материала в отношении характеристик пользователей социальных сетей, который требует систематизации и осмысления.

В социальных сетях часто можно встретить явление именуемое парадоксом дружбы: в среднем друзья любого человека имеют больше друзей, чем он сам. Оно было обнаружено в 1991 году социологом из государственного университета Нью-Йорка Скоттом Фельдом [13].

Для изучения парадокса дружбы следует ввести несколько обозначений. В момент времени t для вершины v_i в графе $G(t) = (V(t), E(t))$ сумма степеней всех соседей v_i равна:

$$s_i(t) = \sum_{j:(v_i, v_j) \in E(t)} \deg_j(t),$$

средняя степень соседей вершины v_i :

$$\alpha_i(t) = \frac{s_i(t)}{\deg_i(t)},$$

а индекс дружбы $\beta_i(t)$ определяется как отношение средней степени соседей v_i к степени самой v_i :

$$\beta_i(t) = \frac{\alpha_i(t)}{\deg_i(t)} = \frac{s_i(t)}{\deg_i^2(t)} = \frac{\sum_{j:(v_i, v_j) \in E(t)} \deg_j(t)}{\deg_i^2(t)}.$$

Таким образом, если средняя степень соседей больше степени v_i и парадокс дружбы выполняется, то $\beta_i(t) > 1$ [14].

Для примера, социальные сети Facebook и Github подтверждают парадокс дружбы, что показано на Рис. 1 [15]. Здесь на оси Oy отложено количество узлов сети, для которых индекс дружбы β_i попадает в диапазон, отложенный на оси Ox . Как видим, значительное большинство вершин графов имеют значение индекса дружбы большее единицы.

Распределение индекса дружбы позволяет выявлять сходства между раз-

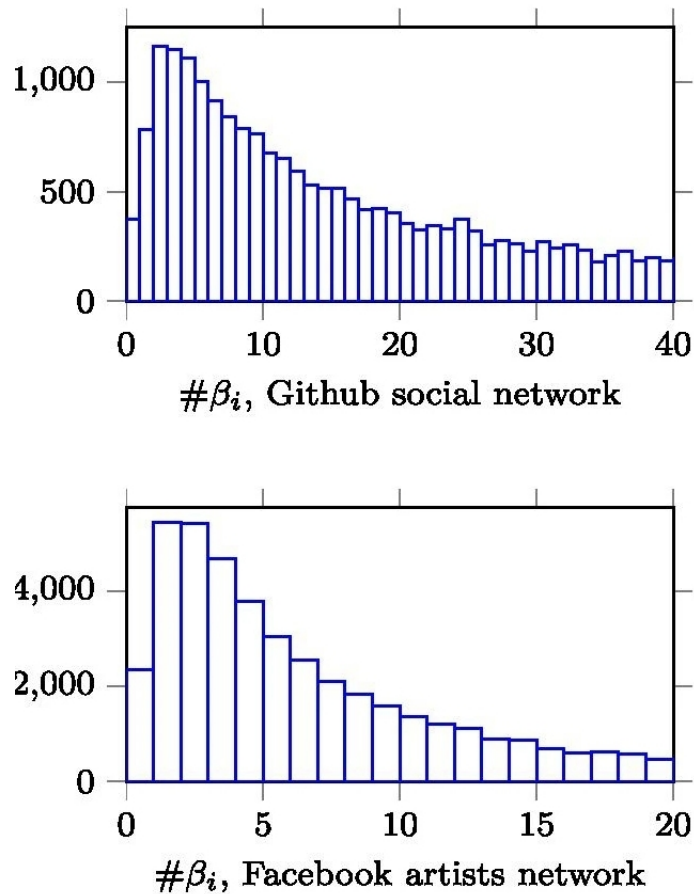


Рисунок 1 – Распределения индекса дружбы в сети телефонных звонков и сети доставки Amazon

личными графами и часто применяется для определения качества модели для симуляции реальной сети. Однако с ростом сети распределение $\Psi_t(k)$ по k масштабируется по мере роста сети, точно так же, как это происходит для значения средних степеней ближайшего соседа [16]. Значения $\Psi_t(k)$ и ANND масштабируются относительно размера сети t , что должно привести к такому эффекту. Следовательно, $\Psi_t(k)$ не является подходящим показателем для сравнения сетей разного размера [17]. Поэтому было бы полезно предложить меру для оценки парадокса дружбы, которая не зависит от размера сети.

1.3 Масштабируемые метрики

На данный момент существует целый ряд различных локальных и глобальных метрик которые не изменяются при изменении размеров графа. Существует несколько подходов к построению масштабируемых метрик.

1.3.1 Химическое расстояние

Учёные из Бостона, Хосе Бенту и Стратис Иоаннидис, в одной из своих работ [18] выделили семейство масштабируемых метрик основанных на химическом расстоянии и расстоянии Шартрана-Кубики-Шульца. В работе рассматривается проблема невозможности сравнения и анализа графов разных размерностей при помощи более часто применимых, немасштабируемых, метрик графов. Данная проблема возникает при решении задач классификации и кластеризации графов, а так же во многих других задачах дата майнинга на графах. Однако авторы статьи смогли выделить две масштабируемых метрики. Первая из них — достаточно хорошо изученная метрика, химическое расстояние. Химическое расстояние $d_{P^n}(A, B)$ между графами G_A и G_B определяется как

$$d_{P^n}(A, B) = \min_{P \in P^n} \|AP - PB\|_F,$$

где A и B - матрицы смежности соответствующих графов $A, B \in \{0, 1\}^{n \times n}$. Вторая — расстояние Шартран-Кубики-Шульца, которое также описывается формулой (1.3.1), однако, отличие заключается в том, что в данном случае A и B отображают матрицы кратчайших путей между вершинами графа. Основным недостатком данного семейства метрик является их высокая вычислительная сложность.

1.3.2 DeepSIM

В статье «DeepSIM: a novel deep learning method for graph similarity computation» [19] была осуществлена попытка решить эти проблемы путём создания модели глубокого обучения, способной определить степень схожести двух графов. Предыдущие попытки применять машинное обучение для визуального анализа графов сталкивались с проблемой неспособности выделить локальные свойства. Для её преодоления была разработана модель DeepSIM, с применением механизма глобально-локального внимания для улучшения работы CNN-модели, оба механизма применяются параллельно и независимо. На Рис. 2 представлена схема устройства модели.

1.3.3 Визуальный анализ

Также существует локально-глобальный подход к сравнению графов разного размера.

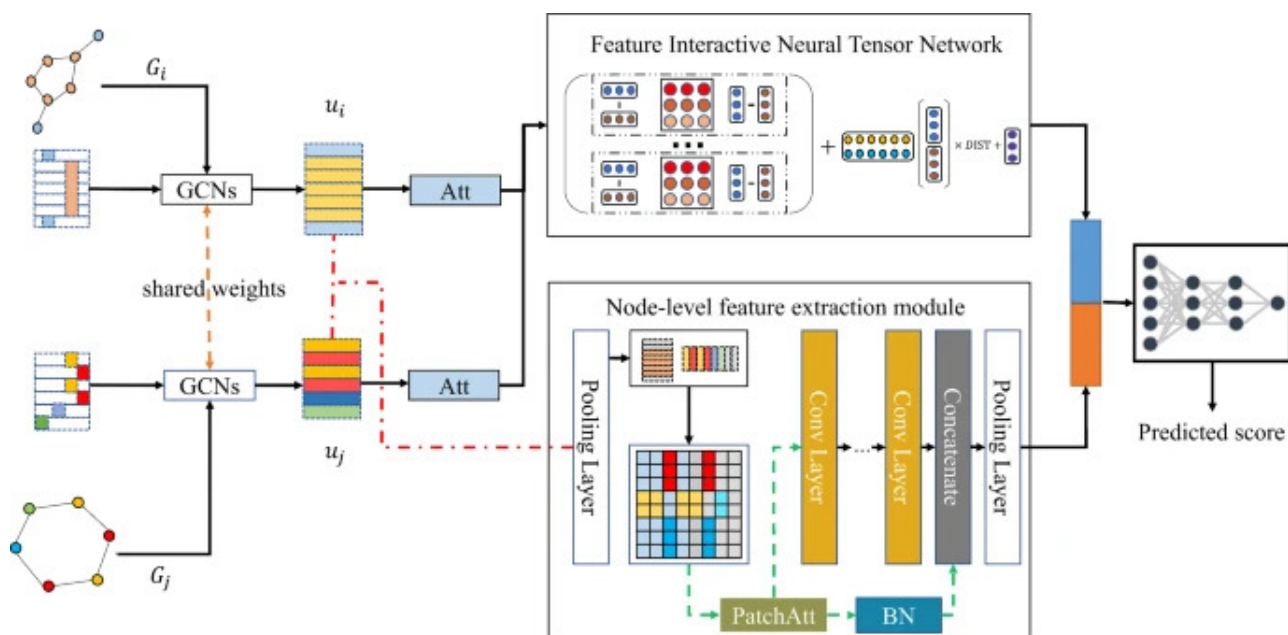


Рисунок 2 – Модель DeepSIM

Так авторы работы «Motif-Based Visual Analysis of Dynamic Networks» [20] предлагают использовать визуальный анализ графов и их подграфов для определения их схожести. В статье рассматриваются сетевые мотивы - это повторяющиеся и статистически значимые подграфы или шаблоны более крупного графика. Сетевые мотивы повторяются в определенной сети или даже среди различных сетей. Каждый из этих подграфов, определяемый определенным шаблоном взаимодействий между вершинами, может отражать структуру, в которой эффективно выполняются определенные функции. Однако обычно мотивы используются только для анализа статических, а не динамических сетей. В данной работе предлагается использовать визуальный анализ сетевых мотивов для сравнения структуры и динамики изменения графов, на Рис. 3 и 4 изображён процесс анализа графов в соответствии с данным методом. В результате работы была показана возможность выделять временные состояния, тенденции и выбросы в динамических сетях. Но у данного метода также имеется ряд недостатков связанных с тем, что данные анализируются человеком вручную, среди них высокая трудоёмкость процесса и определённая субъективность результатов.

1.3.4 ANNR

Помимо этого также существует глобальная масштабируемая метрика именуемая ANNR (Average Nearest Neighbor Rank).

Определение ANNR основывается на ANND (Average Nearest Neighbor



Рисунок 3 – Процесс визуализации сетевых мотивов

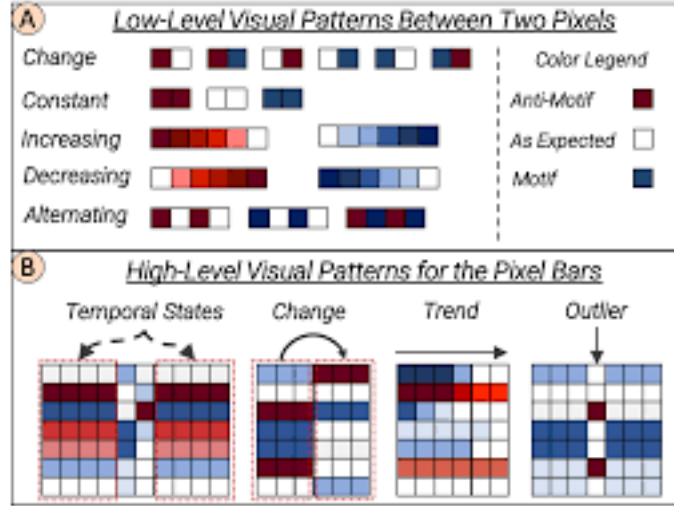


Рисунок 4 – Возможные паттерны среди мотивов

Degree) — глобальном аналоге метрики средней суммы соседей. ANND от k обозначается как $\Phi_n(k)$ и представляет собой среднюю степень соседей всех вершин степени k . Она вычисляется как

$$\Phi(k) = 1_{\{f_n(k) > 0\}} \frac{\sum_{l > 0} h_n(k, l) l}{f_n^*(k)},$$

где $h_n(k, l)$ — совместное распределение степеней узлов на обоих концах случайного ребра графа $G_n(t)$ и $f_n^*(k)$ — эмпирическая плотность степеней, зависящая от размера, определяемые как

$$h_n(k, l) = \frac{1}{L_n(t)} \sum_{i, j: \deg_i(t)=k, \deg_j(t)=l}$$

и

$$f_n^*(k) = \frac{1}{L_n(t)} \sum_{i \rightarrow j} 1_{\{\deg_i(t)=k\}} = \frac{1}{L_n(t)} \sum_{i=1}^n k 1_{\{\deg_i(t)=k\}} = \frac{nk f_n(k)}{L_n(t)}, k = 1, 2, \dots$$

а $L_n(t)$ — это сумма всех степеней

$$L_n(t) = \sum_{i=1}^n \deg_i(t)$$

графа $G_n(t)$ в момент t .

В работе "Average nearest neighbor degrees in scale-free networks" [16] авторы указывают на то, что значения ANND изменяются при росте сети, кроме того, в случае бесконечной дисперсии масштабируемый предел является правильной случайной величиной, которая может изменяться по мере изменения градусной выборки. И предлагают новую глобальную метрику, которую они назвали ANNR (Average Nearest Neighbor Rank). ANNR обозначается как:

$$\Theta_n(k) = 1_{\{f_n(k) > 0\}} \frac{\sum_{l > 0} h_n(k, l) F_n^*(l)}{f_n^*(k)}.$$

Основное отличие ANNR от ANND заключается в том, что ANNR является ранговой величиной, что, в свою очередь достигается благодаря использованию кумулятивного распределения степеней, зависящего от размера:

$$F_n^*(l) = \frac{1}{L_n(t)} \sum_{i=1}^n \deg_i(t) 1_{\{\deg_i(t) \leq l\}}.$$

В ходе выполнения работы были проведены эмпирические эксперименты подтверждающие результаты исследований авторов ANNR. Для этого были построены датасеты по 10 графов содержащих по 10 000 вершин, в соответствии с моделью Барабаши-Альберт и конфигурационной моделью (SCM).

Модель Барабаши—Альберт является одной из первых моделей веб-графов. Веб-граф представляет собой ориентированный мульти-граф, вершинами в котором являются какие-либо конкретные структурные единицы в Интернете: речь может идти о страницах, сайтах, хостах, владельцах и пр. Для определенности будем считать, что вершинами веб-графа служат именно сайты. А рёбрами соединяются вершины, между которыми имеются ссылки.

В своей модели А.-Л. Барабаши и Р. Альберт предложили стратегию предпочтительного присоединения [6]. Её основная идея заключается в том, что вероятность присоединения конкретной вершины ребром к новой вершине пропорциональна степени данной вершины. Здесь и далее степенью вершины $v_i \in V$

графа $G = (V, E)$ называется количество вершин, напрямую связанных с данной, т.е.

$$\deg(v_i) = |\{v \in V : (v, v_i) \in E\}|.$$

Алгоритм формирования сети по модели Барабаши—Альберт заключается в следующем.

1. Первоначально берется полный граф из m вершин, где m — параметр модели.
2. На каждой итерации роста сети добавляется одна новая вершина, которая соединяется m ребрами с уже имеющимися в соответствии с принципом предпочтительного присоединения.

SCM - это статистический ансамбль случайных графов, G имеющих $n = |V(G)|$ вершин помеченных $\{v_j\}_{j=1}^n = V(G)$, создающий распределение вероятностей на G_n (наборе графиков размера n). На ансамбль накладываются n ограничений, а именно, что среднее значение по ансамблю для степени k_j вершины v_j равно заданному значению \widehat{k}_j для всех v_j в $V(G)$. Модель полностью параметризована по своему размеру n и ожидаемой последовательности степеней $\{\widehat{k}_j\}_{j=1}^n$ [21]. Эти ограничения являются как локальными (по одному ограничению, связанному с каждой вершиной), так и мягкими (ограничения на среднее по ансамблю определенных наблюдаемых величин) и, таким образом, дают канонический ансамбль с обширным количеством ограничений. Условия $\langle k_j \rangle = \widehat{k}_j$ накладываются на ансамбль с помощью метода множителей Лагранжа [22].

На Рис.5 можно увидеть сравнение изменений ANND и ANNR при увеличении размера графа. Из данных на графиках можно сделать вывод, что, в отличие от ANND, распределение ANNR действительно остаётся практически неизменным при изменении размера графа.

Полный программный код данного эксперимента содержится в приложении А.

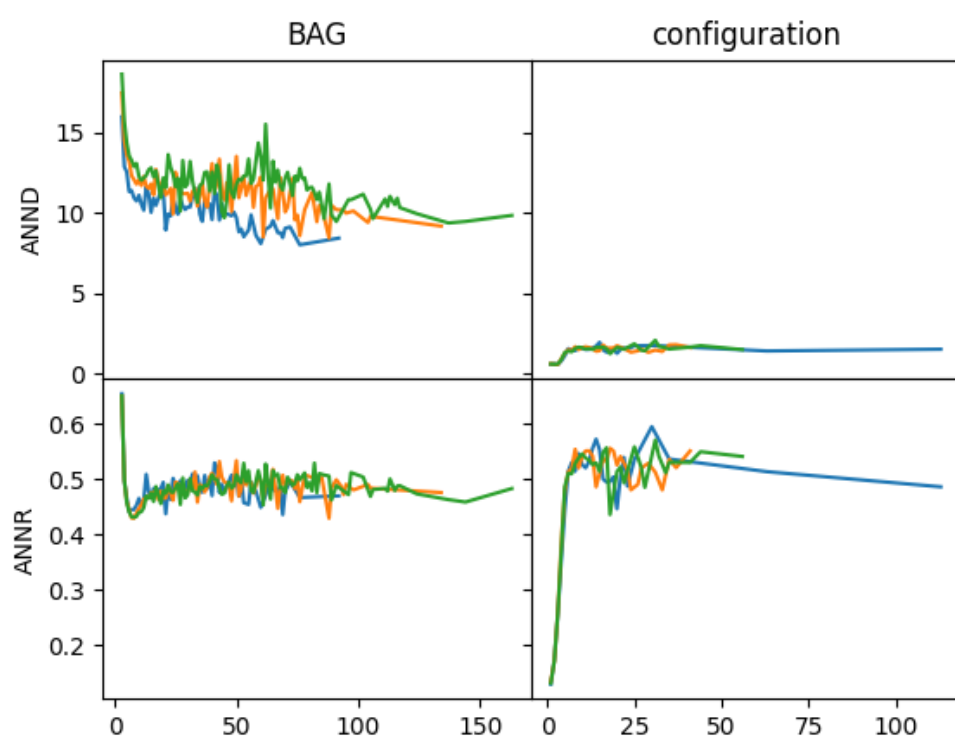


Рисунок 5 – Результаты сравнения ANND и ANNR

2 Модификации индекса дружбы

В данной работе предлагается использование аналогичного подхода для масштабирования и нормализации индекса дружбы, для формирования масштабируемой локальной метрики. ANND ($\Phi_n(k)$) можно рассматривать как глобальный аналог глобальной метрики средней суммы соседей вершины ($\alpha_i(t)$):

$$\begin{aligned}\Phi(k) &= 1_{\{f_n(k)>0\}} \frac{\sum_{l>0} h_n(k, l)l}{f_n^*(k)} = 1_{\{f_n(k)>0\}} \frac{\sum_{l>0} (\frac{1}{L_n} \sum_{i \rightarrow j} 1_{\{deg_i(t)=k, deg_j(t)=l\}})l}{\frac{nk f_n(k)}{L_n}} = \\ &= 1_{\{f_n(k)>0\}} \frac{\frac{\sum_{l>0} (\sum_{i \rightarrow j} 1_{\{deg_i(t)=k, deg_j(t)=l\}})l}{L_n}}{\frac{nk f_n(k)}{L_n}}.\end{aligned}$$

Переобозначив $\sum_{l>0} (\sum_{i \rightarrow j} 1_{\{deg_i(t)=k, deg_j(t)=l\}})$ как $n_{kl}(t)$ и $f_n(k)$ как $n_k(t)$, так как они обозначают количество ребер с вершинами степени k, l и количество узлов степени k , соответственно, а также сократив $\frac{1}{L_n}$ в числителе и знаменателе получим:

$$\Phi(k) = 1_{\{f_n(k)>0\}} \frac{\sum_{l>0} (\sum_{i \rightarrow j} n_{kl}(t))l}{nkn_k(t)}.$$

В свою очередь

$$\alpha_i(t) = \frac{s_i(t)}{deg_i(t)} = \frac{\sum_{j:(v_i, v_j) \in E(t)} deg_j(t)}{deg_i(t)}.$$

В приведённых формулах можно заметить некоторые закономерности: $\sum_{l>0} (\sum_{i \rightarrow j} n_{kl}(t))l$ представляет собой сумму степеней всех соседей всех вершин графа $G_n(t)$ степени k ; а $\sum_{j:(v_i, v_j) \in E(t)} deg_j(t)$ — сумма степеней соседей вершины v_i . Аналогично, выражение $kn_k(t)$ в формуле ANND выполняет ту же роль, что и $deg_i(t)$ в формуле средней суммы соседей — сумму степеней всех интересующих нас вершин (вершин степени k , в случае ANND, и степень вершины v_i , в случае $\alpha_i(t)$). Итак мы имеем:

$$1_{\{f_n(k)>0\}} \frac{\sum_{l>0} (\sum_{i \rightarrow j} n_{kl}(t))l}{nkn_k(t)} \sim \frac{\sum_{j:(v_i, v_j) \in E(t)} deg_j(t)}{deg_i(t)}.$$

Таким образом, можно сделать вывод, что для преобразования средней суммы соседей и индекса дружбы в масштабируемую метрику, как и в случае ANND, достаточно домножить их значения на некий локальный аналог ку-

мулятивного распределения степеней $F_n^*(t)$. В ходе выполнения работы были рассмотрены два варианта такой величины. Их общая формула выглядит как

$$\frac{\sum_{j:(v_i, v_j) \in E(t), \deg_j(t) < \deg_i(t)} \deg_j(t)}{L^*(t)}.$$

Основное отличие между двумя способами заключается в использованном L^* . В первом случае рассматривается глобальное $L_n(t) = \sum_{i=1}^n \deg_i(t)$, а во втором — локальное $L_i(t) = \sum_{j:(v_i, v_j) \in E(t)} \deg_j(t)$.

Предложенная метрика получила обозначение MFI (Modified Friendship Index) и, соответственно, первый вариант обозначен как

$$MFI1_i(t) = \beta_i(t)L_i(t) = \frac{\sum_{j:(v_i, v_j) \in E(t)} \deg_j(t)}{\deg_i^2(t) \sum_{j:(v_i, v_j) \in E(t)} \deg_j(t)},$$

а второй:

$$MFI2_i(t) = \beta_i(t)L_n(t) = \frac{\sum_{j:(v_i, v_j) \in E(t)} \deg_j(t)}{\deg_i^2(t) \sum_{i=1}^n \deg_i(t)}$$

3 Экспериментальная проверка гипотезы

В ходе выполнения работы были проведены эксперименты моделирующие поведение стандартного индекса дружбы и обоих предложенных его модификаций, в сетях случайных графов Барабаши-Альберт и SCM размера 10 000 вершин. Для получения более усреднённых результатов каждый эксперимент проводится 10 раз и в качестве результата берётся их среднее арифметическое.

Результаты экспериментов представлены в виде графиков распределения, с разбиением на 50 групп, перечисленных метрик в различных графах на различных этапах построения сети.

На Рис. 6 можно увидеть результаты описанных экспериментов.

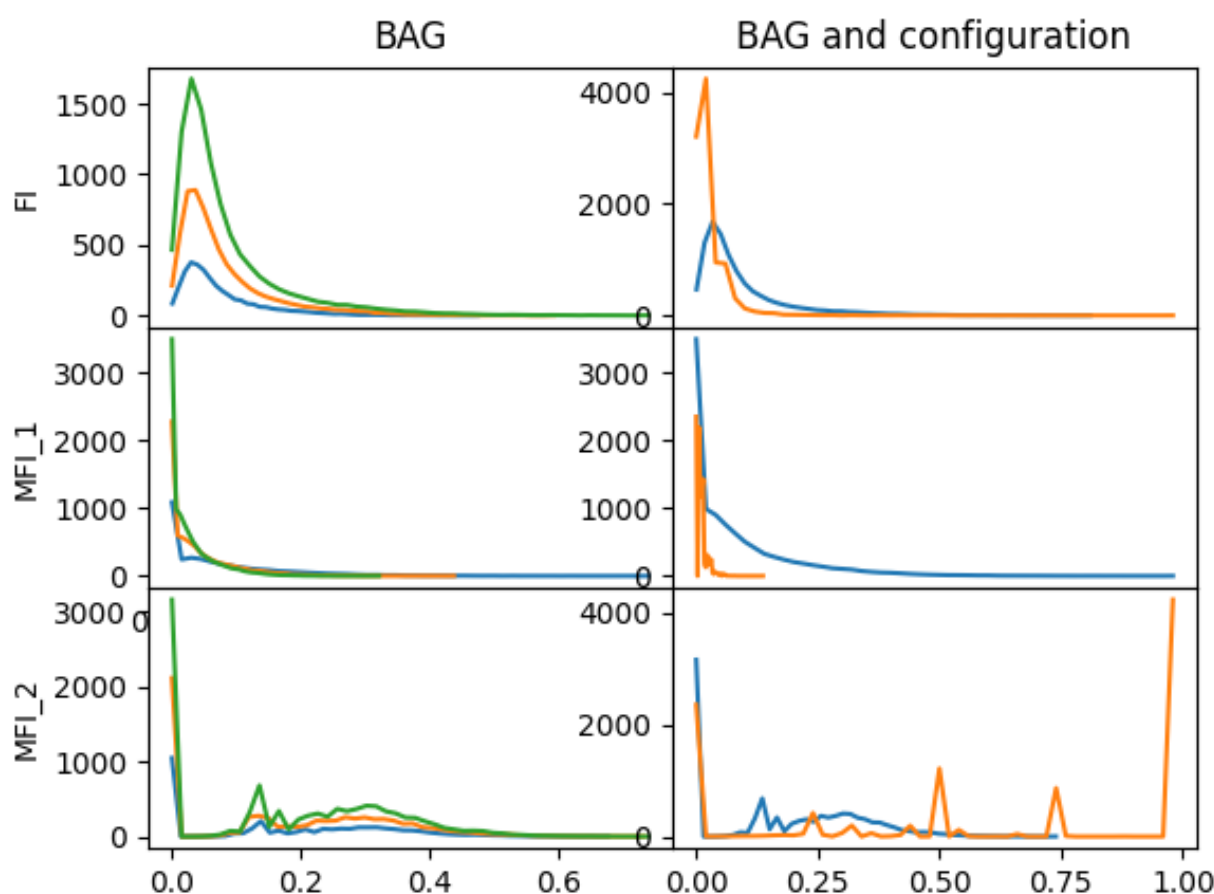


Рисунок 6 – Сравнение поведения $\beta_i(t)$, $MFI1_i(t)$ и $MFI2_i(t)$

Полный текст программы для проведения данного эксперимента представлен в приложении Б.

Также был проведён эксперимент для анализа динамики среднего индекса дружбы, $MFI1_i$ и $MFI2_i$ в сети модели Барабаши-Альберт. Результаты этого

эксперимента также были усреднены по 10 попыткам. Графики изменений средних значений перечисленных величин можно увидеть на Рис. 7.

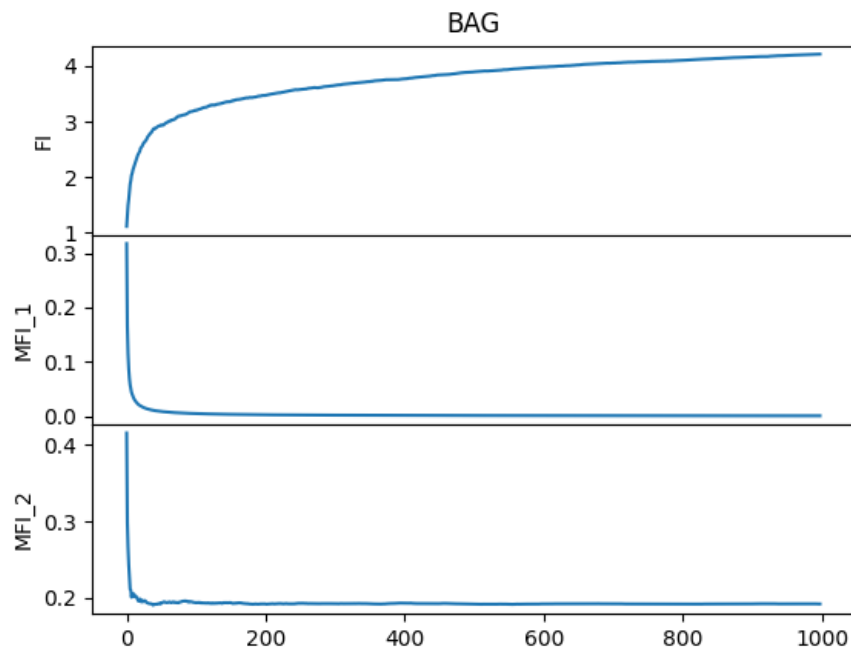


Рисунок 7 – Сравнение динамики изменения $\beta_i(t)$, $MFI1_i(t)$ и $MFI2_i(t)$

4 Анализ результатов

В ходе проведения экспериментов были построены усреднённые графики распределения $\beta_i(t)$, $MFI1_i(t)$ и $MFI2_i(t)$ в графах построенных по модели Барабаши—Альберт с параметром $m = 5$, графики строились трижды, при значениях n принадлежащих множеству $n \in \{3333, 6666, 10\,000\}$. Они представлены в левом столбце Рис. 6. На основании этих графиков можно сделать вывод о том, что $MFI1_i(t) = \beta_i(t)L_i(t)$ является единственной метрикой из рассмотренных, распределение которой остаётся прежним при росте сети, в то время как индекс дружбы и, в меньшей степени, $MFI2_i(t) = \beta_i(t)L_n(t)$ увеличиваются при изменении размера сети.

В правом столбце Рис. 6 расположено сравнение графиков распределения соответствующих метрик в графах одинакового размера построенных по разным моделям: синие графики отражают распределение метрик в сети Барабаши—Альберт при $n = 10\,000$, а оранжевые графики — распределение в графе конфигурационной модели (SCW) построенной по последовательности Парето при значении параметра $\lambda = 2.5$ и $n = 10\,000$. Эти графики указывают на то, что не смотря на способность масштабирования при росте сети $MFI1_i(t)$ позволяет различать графы построенные по разным моделям и, соответственно, обладающих различными свойствами.

Из графиков динамики средних значений $\beta_i(t)$, $MFI1_i(t)$ и $MFI2_i(t)$ представленных на Рис. 7 можно сделать вывод, о том что в отличии от бесконечно растущего индекса дружбы, средние значения обеих предложенных метрик весьма быстро стабилизируются после короткого периода быстрого изменения среднего значения на начальных итерациях генерации графа. Это можно объяснить, тем что на начальных итерациях структура графа модели Барабаши—Альберт непредсказуема и претерпевает значительные изменения при каждом добавлении новой вершины, однако со временем модель стабилизируется и её поведение становится всё более и более предсказуемым.

Также из Рис. 8, на котором изображены результаты того же эксперимента на логарифмической шкале, можно заметить, что график $MFI1_i(t)$ представляет собой нисходящую прямую линию, что указывает на то что среднее значение данной метрики изменяется по обратному степенному закону.

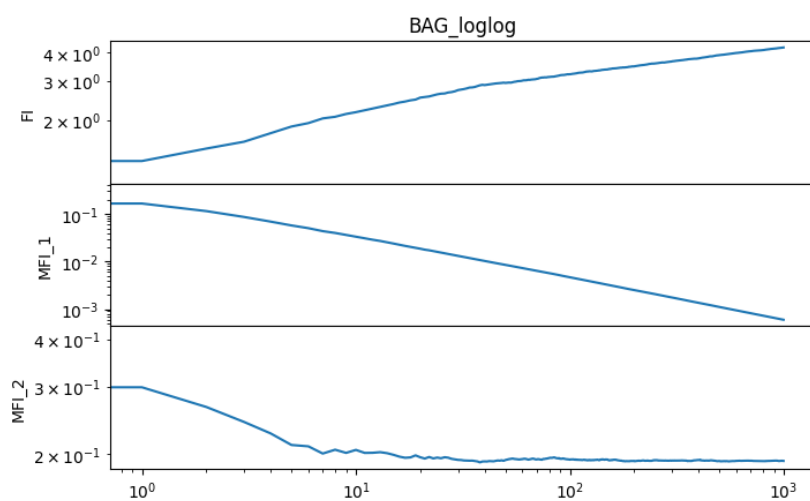


Рисунок 8 – Сравнение динамики изменения $\beta_i(t)$, $MFI1_i(t)$ и $MFI2_i(t)$ на логарифмической шкале

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы были рассмотрены различные локальные и глобальные метрики графов. А также был предложен вариант масштабируемой локальной метрики, потенциально позволяющей сравнивать графы разного размера и были проведены эксперименты подтверждающие данное свойство этой метрики.

Однако, не смотря экспериментальное подтверждение масштабируемости метрики $MFI1_i(t)$, наличие этого свойства не было доказано математически, а следовательно, без дополнительных исследований невозможно утверждать, что $MFI1_i(t)$ является масштабируемой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Erdős, P. On random graphs i / P. Erdős, A. Rényi // *Publicationes Mathematicae Debrecen*. — 1959. — Vol. 6. — Pp. 290–297.
- 2 Gilbert, E. N. Random graphs / E. N. Gilbert // *Annals of Mathematical Statistics*. — 1959. — Vol. 30, no. 4. — Pp. 1141–1144.
- 3 Blum, A. Foundations of data science. — 2020. <https://api.semanticscholar.org/CorpusID:123439379>.
- 4 Bollobas, B. Random Graphs / B. Bollobas. Cambridge Studies in Advanced Mathematics. — 2 edition. — Cambridge University Press, 2001.
- 5 Sirocchi, C. Topological network features determine convergence rate of distributed average algorithms / C. Sirocchi, A. Bogliolo // *Scientific Reports*. — Dec 2022. — Vol. 12, no. 1. — P. 21831. <https://doi.org/10.1038/s41598-022-25974-w>.
- 6 Albert, R. Statistical mechanics of complex networks / R. Albert, A.-L. Barabasi // *Reviews of Modern Physics*. — 2002. — Vol. 74, no. 1. — Pp. 47 – 98.
- 7 Райгородский, А. М. Модели случайных графов и их применени / А. М. Райгородский. — Москва, М.: Издательство МЦНМО, 2011.
- 8 Райгородский, А. М. Модели случайных графов / А. М. Райгородский // *Труды МФТИ*. — 2010. — Т. 2, № 4. — С. 130 – 140.
- 9 Берновски, М. Случайные графы, модели и генераторы безмасштабных графов. / М. Берновски, Н. Кузюрин // *Труды Института системного программирования РАН*. — 2012. — Т. 22, № 4. — С. 419 – 432.
- 10 David, E. Networks, Crowds, and Markets: Reasoning About a Highly Connected World / E. David, K. Jon. — USA: Cambridge University Press, 2010.
- 11 Longa, A. Neighbourhood matching creates realistic surrogate temporal networks / A. Longa, G. Cencetti, S. Lehmann, A. Passerini, B. Lepri // *arXiv preprint arXiv:2205.08820*. — 2022.
- 12 Пользователи социальных сетей: современные исследования [Электронный ресурс]. — URL: <https://psychojournal.ru/article/887-polzovateli-socialnyh-setey-sovremennye-issledovaniya.html> (Дата обращения 10.03.2023). Загл. с экр. Яз. рус.

- 13 *Feld, S. L.* Why your friends have more friends than you do / S. L. Feld // *American Journal of Sociology*. — 1991. — Vol. 96. — Pp. 1464 – 1477.
- 14 *Pal, S.* A study on the friendship paradox quantitative analysis and relationship with assortative mixing / S. Pal, F. Yu, Y. Novick, A. Swami, A. Bar-Noy // *Applied Network Science*. — 09 2019. — Vol. 4. — Pp. 71 – 118.
- 15 *Sidorov, S.* Friendship paradox in growth networks: analytical and empirical analysis / S. Sidorov, S. Mironov, A. Grigoriev // *Appl Netw Sci*. — 2023. — Vol. 74, no. 6. — Pp. 47 – 51.
- 16 *Yao, D.* Average nearest neighbor degrees in scale-free networks / D. Yao, P. van der Hoorn, N. Litvak. — 2017. <https://arxiv.org/abs/1704.05707>.
- 17
- 18 *Bento, J.* A family of tractable graph metrics / J. Bento, S. Ioannidis // *Applied Network Science*. — Nov 2019. — Vol. 4, no. 1. — P. 107. <https://doi.org/10.1007/s41109-019-0219-z>.
- 19 *Liu, B.* Deepsim: a novel deep learning method for graph similarity computation / B. Liu, Z. Wang, J. Zhang, J. Wu, G. Qu // *Soft Computing*. — Jan 2024. — Vol. 28, no. 1. — Pp. 61–76. <https://doi.org/10.1007/s00500-023-09288-1>.
- 20 *Cakmak, E.* Motif-based visual analysis of dynamic networks // 2022 IEEE Visualization in Data Science (VDS). — 2022. — Pp. 17–26.
- 21 *Garlaschelli, D.* Covariance structure behind breaking of ensemble equivalence in random graphs / D. Garlaschelli, F. den Hollander, A. Roccaverde // *Journal of Statistical Physics*. — 2018. — Vol. 173, no. 3–4. — P. 644–662.
- 22 *Park, J.* Statistical mechanics of networks / J. Park, M. E. J. Newman // *Physical Review E*. — 2004. — Vol. 70, no. 6.

ПРИЛОЖЕНИЕ А

Текст программы для проведения эксперимента по сравнению ANND и ANNR

В этом приложении содержится текст программы для проведения эксперимента по сравнению ANND и ANNR.

```
1 from diploma import d, neighbours, index, my_bag, run, s
2 import matplotlib.pyplot as plt
3 import math
4 import networkx as nx
5 import numpy as np
6
7 n = 1000
8 graphCount = 30
9
10 def measure_kvise (degrees, neighbours, f):
11     byDegree = {}
12     for i in range(len(degrees)) :
13         if degrees[i] in byDegree:
14             byDegree[degrees[i]].append(i)
15         else:
16             byDegree[degrees[i]] = [i]
17
18     ans = {}
19     for (k, vertexes) in byDegree.items():
20         ans[k] = (f(degrees, neighbours, vertexes, k))
21     return ans
22
23 def ANND_helper (degrees, neighbours, vertexes, k):
24     if len(vertexes) == 0:
25         return 0
26
27     L = sum(degrees)
28     f = len(vertexes) / len(degrees)
29     fs = len(degrees) * k * f / L
30
31     h = {}
32     for v in vertexes:
33         for nbr in neighbours[v]:
34             if degrees[nbr] in h:
```



```

35         h[degrees[nbr]] += 1
36     else:
37         h[degrees[nbr]] = 1
38     if k in h:
39         h[k] -= len(vertexes) # to fix duplicating edges
40     sumhl = 0
41     for (key, val) in h.items():
42         sumhl += key * val / L
43
44     return sumhl / fs
45
46 def ANND(degrees, neighbours): return measure_kvise(degrees, neighbours,
47     ↪ ANND_helper)
48
49 def ANND2_helper (degrees, neighbours, vertexes, k):
50     S = s(degrees, neighbours)
51     sum = 0
52     for v in vertexes:
53         sum += S[v]
54     return sum / len(vertexes)
55
56 def ANND2(degrees, neighbours): return measure_kvise(degrees, neighbours,
57     ↪ ANND2_helper)
58
59 def ANNR_helper (degrees, neighbours, vertexes, k):
60     if len(vertexes) == 0:
61         return 0
62
63     L = sum(degrees)
64     f = len(vertexes) / len(degrees)
65     fs = len(degrees) * k * f / L
66
67     h = {}
68     for v in vertexes:
69         for nbr in neighbours[v]:
70             if degrees[nbr] in h:
71                 h[degrees[nbr]] += 1
72             else:
73                 h[degrees[nbr]] = 1
74
75     if k in h:
76         h[k] -= len(vertexes) # to fix duplicating edges

```

```

74     sumhFs = 0
75     for (key, val) in h.items():
76         sumD = 0
77         for d in degrees:
78             if d <= key:
79                 sumD += d
80         sumFs = sumD / L
81         sumhFs += sumFs * val / L
82
83     res = sumhFs / fs
84
85     return res
86
87 def ANNR(degrees, neighbours): return measure_kvise(degrees, neighbours,
↪ ANNR_helper)
88
89 def AFR_helper (degrees, neighbours, vertexes, k):
90     return ANNR_helper(degrees, neighbours, vertexes, k) / k
91 def AFR(degrees, neighbours): return measure_kvise(degrees, neighbours,
↪ AFR_helper)
92
93 if __name__ == '__main__':
94     # for m in [3, 5]:
95         # for p in [0.25, 0.5, 0.75]:
96     m = 3
97     p = 0.25
98
99     fig = plt.figure()
100     gs = fig.add_gridspec(2, 2, hspace=0, wspace=0)
101     ax1, ax2= gs.subplots(sharex='col', sharey='row')
102     graphs = run(graphCount, 6, my_bag, n, [m, p], [ANND, ANNR, AFR], [n //
↪ 3, n // 3, n // 3])
103
104     ax1[0].set_title("BAG")
105     ax1[0].set_ylabel("ANND")
106     mean = [{ } for _ in graphs[0][0]]
107     count = [{ } for _ in graphs[0][0]]
108     for graph in graphs[1:]:
109         for (step, curMean, curCount) in zip(graph[0], mean, count):
110             for k in step.keys():
111                 if not k in curMean.keys():

```

```

112         curMean[k] = 0
113         curCount[k] = 0
114         curMean[k] += step[k]
115         curCount[k] += 1
116     for (curMean, curCount) in zip(mean, count):
117         for k in curMean.keys():
118             curMean[k] /= curCount[k]
119     for curMean in mean:
120         lst = sorted(curMean.items())
121         x, y = zip(*lst)
122         ax1[0].plot(x, y)
123
124     ax2[0].set_ylabel("ANNR")
125     mean = [{ } for _ in graphs[0][1]]
126     count = [{ } for _ in graphs[0][1]]
127     for graph in graphs[1:]:
128         for (step, curMean, curCount) in zip(graph[1], mean, count):
129             for k in step.keys():
130                 if not k in curMean.keys():
131                     curMean[k] = 0
132                     curCount[k] = 0
133                     curMean[k] += step[k]
134                     curCount[k] += 1
135     for (curMean, curCount) in zip(mean, count):
136         for k in curMean.keys():
137             curMean[k] /= curCount[k]
138     for curMean in mean:
139         lst = sorted(curMean.items())
140         x, y = zip(*lst)
141         ax2[0].plot(x, y)
142
143     # ax3[0].set_ylabel("AFR")
144     # mean = [{ } for _ in graphs[0][2]]
145     # count = [{ } for _ in graphs[0][2]]
146     # for graph in graphs[1:]:
147     #     for (step, curMean, curCount) in zip(graph[2], mean, count):
148     #         for k in step.keys():
149     #             if not k in curMean.keys():
150     #                 curMean[k] = 0
151     #                 curCount[k] = 0
152     #                 curMean[k] += step[k]

```

```

153     #             curCount[k] += 1
154     # for (curMean, curCount) in zip(mean, count):
155     #     for k in curMean.keys():
156     #         curMean[k] /= curCount[k]
157     # for curMean in mean:
158     #     lst = sorted(curMean.items())
159     #     x, y = zip(*lst)
160     #     ax3[0].plot(x, y)
161
162
163
164     steps = [int(n / 3), int(2 * n / 3), n]
165     for step in steps:
166         meanANND = {}
167         meanANNR = {}
168         meanAFR = {}
169         countANND = {}
170         countANNR = {}
171         countAFR = {}
172         for _ in range(graphCount):
173             sequence = list(np.rint((np.random.pareto(2.5, n) + 1)))
174             if not(nx.algorithms.is_multigraphical(sequence)):
175                 sequence[0] += 1
176             M = nx.configuration_model(sequence)
177             G = nx.Graph()
178             for u,v in M.edges():
179                 if not(G.has_edge(u,v)):
180                     G.add_edge(u, v)
181             annr = ANNRR(list(list(zip(*G.degree))[1]), [list(nbrs.keys()) for
↪ _, nbrs in G.adjacency()])
182             for k in annr.keys():
183                 if not k in meanANNR.keys():
184                     meanANNR[k] = 0
185                     countANNR[k] = 0
186                     meanANNR[k] += annr[k]
187                     countANNR[k] += 1
188             annd = ANND(list(list(zip(*G.degree))[1]), [list(nbrs.keys()) for
↪ _, nbrs in G.adjacency()])
189             for k in annd.keys():
190                 if not k in meanANND.keys():
191                     meanANND[k] = 0

```

```

192         countANND[k] = 0
193         meanANND[k] += annd[k]
194         countANND[k] += 1
195     afr = AFR(list(list(zip(*G.degree))[1]), [list(nbrs.keys()) for
↪     _, nbrs in G.adjacency()])
196     for k in afr.keys():
197         if not k in meanAFR.keys():
198             meanAFR[k] = 0
199             countAFR[k] = 0
200             meanAFR[k] += afr[k]
201             countAFR[k] += 1
202
203     for i in meanANND.keys():
204         meanANND[i] /= countANND[i]
205     for i in meanANNR.keys():
206         meanANNR[i] /= countANNR[i]
207     for i in meanAFR.keys():
208         meanAFR[i] /= countAFR[i]
209     # print(meanANND)
210     # print(meanANNR)
211
212     ax1[1].set_title("configuration")
213     # ax1[1].set_ylabel("ANND")
214     x, y = zip(*sorted(meanANND.items()))
215     ax1[1].plot(x, y)
216     # ax2[1].set_title("ANNR")
217     x, y = zip(*sorted(meanANNR.items()))
218     ax2[1].plot(x, y)
219     # x, y = zip(*sorted(meanAFR.items()))
220     # ax3[1].plot(x, y)
221
222
223 plt.show()

```

ПРИЛОЖЕНИЕ Б

Текст программы для проведения эксперимента по сравнению индекса дружбы и его модификаций

В этом приложении содержится текст программы для проведения эксперимента по сравнению стандартного индекса и его двух модификаций: MFI1 и MFI2.

```
1 from diploma import d, neighbours, index, my_bag, run, s, beta
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import networkx as nx
5
6 def mfi1 (degrees, neighbours):
7     L = sum(degrees)
8     sumLTD = [] # sum of degrees Less Than Degree
9     for (dgr, nbrs) in zip(degrees, neighbours):
10         sumLTD.append(sum([degrees[nbr] if degrees[nbr] <= dgr else 0 for nbr
11             ↪ in nbrs]))
12     return [bi * LTD / L for (bi, LTD) in zip(beta(degrees, neighbours),
13         ↪ sumLTD)]
14
15 def mfi2 (degrees, neighbours):
16     sumLTD = [] # sum of degrees Less Than Degree
17     for (dgr, nbrs) in zip(degrees, neighbours):
18         sumLTD.append(sum([degrees[nbr] if degrees[nbr] <= dgr else 0 for nbr
19             ↪ in nbrs]))
20     ans = []
21     for (bi, LTD, nbrs) in zip(beta(degrees, neighbours), sumLTD, neighbours):
22         L = sum(degrees[nbr] for nbr in nbrs)
23         ans.append(bi * LTD / L)
24     return ans
25
26 if __name__ == "__main__":
27     n = 10000
28     m = 5
29     p = 0.25
30     graphCount = 10
31     histBins = 50
32     steps = [n // 3, 2 * n // 3, n]
```

```

31 fig = plt.figure()
32 gs = fig.add_gridspec(3, 2, hspace=0, wspace=0)
33 ax1, ax2, ax3 = gs.subplots()
34 graphs = run(graphCount, 6, my_bag, n, [m, p], [beta, mfi1, mfi2], [n //
↪ 3, n // 3, n // 3])
35
36 ax1[0].set_title("BAG")
37 ax1[1].set_title("BAG and configuration")
38
39 ax1[0].set_ylabel("FI")
40 for i in range(len(graphs[0][0])):
41     meanCount = np.zeros(shape = histBins)
42     meanBeta = np.zeros(shape = histBins + 1)
43     for graph in graphs:
44         hist = np.histogram(graph[0][i], bins=histBins)
45         meanCount += hist[0]
46         meanBeta += hist[1]
47     meanCount /= graphCount
48     meanBeta /= graphCount
49     ax1[0].plot(meanBeta[:-1], meanCount)
50 ax1[1].plot(meanBeta[:-1], meanCount)
51
52 ax2[0].set_ylabel("MFI_1")
53 for i in range(len(graphs[1][0])):
54     meanCount = np.zeros(shape = histBins)
55     meanMFI = np.zeros(shape = histBins + 1)
56     for graph in graphs:
57         hist = np.histogram(graph[1][i], bins=histBins)
58         meanCount += hist[0]
59         meanMFI += hist[1]
60     meanCount /= graphCount
61     meanMFI /= graphCount
62     ax2[0].plot(meanMFI[:-1], meanCount)
63 ax2[1].plot(meanMFI[:-1], meanCount)
64
65 ax3[0].set_ylabel("MFI_2")
66 for i in range(len(graphs[2][0])):
67     meanCount = np.zeros(shape = histBins)
68     meanMFI = np.zeros(shape = histBins + 1)
69     for graph in graphs:
70         hist = np.histogram(graph[2][i], bins=histBins)

```

```

71         meanCount += hist[0]
72         meanMFI += hist[1]
73     meanCount /= graphCount
74     meanMFI /= graphCount
75     ax3[0].plot(meanMFI[:-1], meanCount)
76 ax3[1].plot(meanMFI[:-1], meanCount)
77
78 meanBetaCount = np.zeros(shape = histBins)
79 meanMFI1Count = np.zeros(shape = histBins)
80 meanMFI2Count = np.zeros(shape = histBins)
81 meanBeta = np.zeros(shape = histBins + 1)
82 meanMFI1 = np.zeros(shape = histBins + 1)
83 meanMFI2 = np.zeros(shape = histBins + 1)
84 for _ in range(graphCount):
85     sequence = list(np.rint((np.random.pareto(2.5, n) + 1)))
86     if not(nx.algorithms.is_multigraphical(sequence)):
87         sequence[0] += 1
88     M = nx.configuration_model(sequence)
89     G = nx.Graph()
90     for u,v in M.edges():
91         if not(G.has_edge(u,v)):
92             G.add_edge(u, v)
93     betaData = np.histogram(beta(list(list(zip(*G.degree))[1]),
94     ↪ [list(nbrs.keys()) for _, nbrs in G.adjacency()])), bins=histBins)
95     meanBetaCount += betaData[0]
96     meanBeta += betaData[1]
97     mfi1Data = np.histogram(mfi1(list(list(zip(*G.degree))[1]),
98     ↪ [list(nbrs.keys()) for _, nbrs in G.adjacency()])), bins=histBins)
99     meanMFI1Count += mfi1Data[0]
100     meanMFI1 += mfi1Data[1]
101     mfi2Data = np.histogram(mfi2(list(list(zip(*G.degree))[1]),
102     ↪ [list(nbrs.keys()) for _, nbrs in G.adjacency()])), bins=histBins)
103     meanMFI2Count += mfi2Data[0]
104     meanMFI2 += mfi2Data[1]
105 meanBetaCount /= graphCount
106 meanBeta /= graphCount
107 ax1[1].plot(meanBeta[:-1], meanBetaCount)
108 meanMFI1Count /= graphCount
109 meanMFI1 /= graphCount
110 ax2[1].plot(meanMFI1[:-1], meanMFI1Count)
111 meanMFI2Count /= graphCount

```



```
109     meanMFI2 /= graphCount
110     ax3[1].plot(meanMFI2[:-1], meanMFI2Count)
111
112     plt.show()
```