ЕРЕВАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Оганесян Минас Генрикович

ЦЕЛЕВОЕ ПРОЕКТИРОВАНИЕ СЕТЕВЫХ СИСТЕМ МНОГОМАСШТАБНОЙ АРХИТЕКТУРЫ С ИСПОЛЬЗОВАНИЕМ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ

Диссертация

на соискание учёной степени кандидата технических наук по специальности 05.13.04 “Математическое и программное обеспечение вычислительных машин, комплексов, систем и сетей»

Научный руководитель:

доктор физико-математических наук,

профессор Нигиян С.А.

Ереван - 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. Теория сетей является активно развивающейся областью науки. Множество объектов и явлений представляют в виде статических и динамических сетевых структур, что позволяет применять аппарат теории сетей для широкого спектра явлений и задач. Большой интерес вызывают критические явления в сетях, происходящие в результате различных процессов, ввиду наблюдаемых аналогий с различными физическими, химическими и биологическими явлениями. Важным случаем процесса, приводящего к подобному явлению – фазовому переходу – является процесс направленного развития сетей с топологическими ограничениями Изучение упомянутых явлений невозможно осуществить в полной мере без их программной реализации и анализа экспериментальных результатов, полученных на основе компьютерных симуляций. Эти результаты необходимо получать для различных значений параметров рассматриваемых явлений; необходимо рассматривать ансамбли сетей, которые включают тысячи сетей с одинаковыми параметрами; кроме того, само явление может требовать проведения множества шагов, количество которых достигает десятков миллионов. Поэтому, программная реализация требует применения различных техник распределения и распараллеливания выполняемых вычислений. Множество существующих программных систем моделирования сетей не позволяют рассматривать динамические процессы на сетях, а предназначены для их статического анализа и визуализации. Системы, позволяющие моделировать динамические процессы в сетях, являются узконаправленными и предназначены для проведения исследований в специфичных областях, таких, как, например, сети протеиновых взаимодействий или компьютерные сети. Другие системы предоставляют общие подходы для моделирования подобных процессов, однако от пользователя требуется полное описание необходимых действий, что требует много времени и соответствующих знаний среды от пользователя для моделирования конкретного процесса и не обеспечивает гибкости решения относительно модификаций процесса. Кроме того, результаты экспериментов требуют соответствующей автоматической обработки с целью получения необходимых характеристик и динамик их изменения, что также необходимо реализовать пользователю. Следовательно, актуальной задачей является создание программной системы, которая позволит выполнять необходимое моделирование рассматриваемой динамики с возможностью легкой и быстрой настройки под конкретные характеристики и последующего автоматического анализа полученных экспериментальных данных.

Объект и предмет исследования. Основными объектами исследования данной работы являются:

1. Алгоритмы и программные методы моделирования динамики направленного развития случайных сетей.
2. Параллельные и распределенные технологии моделирования.
3. Динамика направленного развития сложных сетевых структур.
4. Критические и коллективные явления в сетевых структурах.

Основными предметами исследования представленной работы являются:

1. Применение параллельных и распределенных технологий для эффективного компьютерного моделирования сетей и процесса направленного развития в них.
2. Исследование поведения сетевых структур и их характеристик при направленном развитии.
3. Исследование фазового перехода при рассматриваемой динамике.

Цель и задачи исследования. Основной целью диссертационной работы является разработка алгоритмов и программных методов для эффективного компьютерного моделирования направленных процессов в сетях и их применение для экспериментального исследования процесса направленного развития сетей.

Исходя из указанной цели, в работе поставлены и решены следующие задачи:

1. Разработка алгоритмов и программных методов для моделирования процесса направленного развития сетей.
2. Разработка программных средств для моделирования процесса направленного развития сетей с возможностью получения строения и основных свойств рассматриваемых сетей.
3. Моделирование динамики развития сетей с использованием разработанной системы; получение и анализ экспериментальных данных.
4. Исследование процесса направленного развития сетевых структур и характеристик изучаемых и конструируемых сетей.

Научная новизна. Разработаны алгоритмы и программные методы моделирования направленных процессов в сетях. На их основе разработана программная система, позволившая получить экспериментальные результаты и выявить приведенные ниже новые явления, наблюдае-

мые при моделировании направленного развития случайных сетей Эрдёша-Реньи с топологическими ограничениями:

1. Процесс направленного развития с сохранением степеней вершин приводит к расщеплению сети на плотные (почти полные) компоненты со слабыми связями между компонентами. Данное явление называется структурообразованием.
2. Расщепление сопровождается образованием двух зон в спектральной плотности матриц смежности рассматриваемого ансамбля сетей. Основная зона в процессе развития сужается и меняет форму от Вигнеровской в начале к треугольной в конце рассматриваемого процесса. Вторая зона в начале состоит из нескольких изолированных значений, а в конце процесса зона становится плотной и связанной.
3. Матрицы смежности сетей изменяют свою структуру от случайных в начале рассматриваемого процесса к блочно-диагональным в конце. Матрицы смежности финальных сетей содержат блоки примерно равных размеров, количество которых обратно пропорционально вероятности связей между вершинами сети.

Практическая ценность. Разработанная программная система может быть использована для разностороннего исследования различных направленных динамик в сетях. Экспериментальные результаты исследований, проведенных на случайных сетях Эрдёша-Реньи, могут быть использованы для описания различных физических и биологических явлений, таких, как явление почкования в мембранах липидов, некоторых явлений из теории струн и квантовой гравитации, а также для описания процессов в нейронных сетях мозга. Процесс направленного развития сетей с топологическими ограничениями можно применять как инструмент для разбиения случайных сетей на оптимальные части из почти полных подсетей.

На защиту выносятся следующие положения:

1. Разработаны алгоритмы и программные методы моделирования процесса направленного развития сетей. Разработана программная система, в которой реализованы соответствующие алгоритмы с применением программных методов. Система, кроме моделирования процесса направленного развития сетей, позволяет проводить разносторонний анализ сетей и получение динамик различных свойств. Система показала свою эффективность в ходе проведенных экспериментальных исследований и внедрена в Институте химической физики им. Н.Н. Семенова РАН.

2. Случайная сеть в результате процесса направленного развития с топологическими ограничениями расщепляется на множество почти полных кластеров (кликов).
3. Количество кластеров, образовывающихся в результате процесса направленного развития случайных сетей Эрдёша-Реньи, обратно пропорционально вероятности связи.
4. Слабые связи между кластерами, образовывающимися в результате процесса направленного развития случайных сетей Эрдёша-Реньи, определяют масштабно-инвариантные свойства финальных сетей.

Апробация результатов работы. Основные результаты диссертации были представлены на семинаре кафедры программирования и информационных технологий ЕГУ, на общем семинаре факультета информатики и прикладной математики ЕГУ, на научной школе-семинаре "Технологии моделирования сложных физических и биологических систем" (Цмакаог, 2015), на международной конференции Computer Science and Information Technologies - 2015 (CSIT, Ереван, 2015) и на международном семинаре Workshop on Critical and collective effects in graphs and networks (CCEGN, Москва, 2016).

Публикации. Основные результаты диссертации опубликованы в 3 публикациях ([1], [2], [3]).

Структура и объем диссертации. Диссертация состоит из введения, трех глав и заключения общим объёмом 110 страниц, включая 94 наименования литературы. Работа включает 24 рисунка, 3 диаграммы и 3 таблицы.

Во введении представлен общий обзор области исследований, актуальность рассматриваемых задач и приведены основные результаты диссертационной работы.

В первой главе представлен обзор области исследований. Представлена теория сетей, основные рассматриваемые задачи и имеющиеся результаты. Представляется процесс направленного развития сетей, который является объектом исследования и обосновывается актуальность его исследования. Далее представлены вопросы программного моделирования процессов в сетях, доступные системы и библиотеки моделирования сетей и приводятся их основные недостатки в контексте программного моделирования рассматриваемой задачи. Далее формулируются объекты и предметы исследований, представляются поставленные цели и задачи исследований и описываются методология и методы исследований.

Во второй главе представлены основные теоретические и технологические результаты и идеи, лежащие в основе диссертационной работы. Глава состоит из четырех разделов. Раздел 2.1 описывает модель случайных графов Эрдёша-Реньи. Описываются основные свойства сетей, методы визуализации сетей, представлены способы обнаружения сообществ, а также спектральные свойства матриц. Раздел 2.2 представляет идею семплирования и дает описание одного из его распространенных вариантов – алгоритма Метрополиса – Гастингса. Данная идея лежит в основе процесса направленного развития сетей, рассматриваемого в данной работе, ввиду того, что позволяет выполнить структурно-параметрическую идентификацию наилучшей статистической модели случайного процесса и является эффективным приемом для логичного смыслового увязывания статистических свойств выборки и цели моделирования. Раздел 2.3 описывает модель процесса направленного развития сетей. Раздел 2.4 содержит описание основных технологий параллельного и распределенного программирования, рассмотренных при разработке программной среды моделирования.

В третьей главе описывает алгоритмы и программные методы, разработанные для моделирования направленных процессов в сетях. Представлена программная система, разработанная для проведения необходимых исследований. Глава состоит из семи разделов. Раздел 3.1 представляет основные требования к программной системе для проведения исследований. В разделе 3.2 приведено обоснование необходимости проектирования и создания специализированной программной системы с использования параллельных и распределенных технологий. Раздел 3.3 описывает детальный анализ вопросов применения некоторых параллельных и распределенных технологий, в частности обосновывается применение технологии MPI. Раздел 3.4 содержит описание разработанных алгоритмов моделирования процесса эволюции для двух рассмотренных вариантов переключения связей. Далее представлены ключевые детали процесса реализации, наладки и оптимизации различных фрагментов (раздел 3.5); представлена архитектуры системы, описаны её уровни и составляющие модули и механизмы взаимодействия между ними. Детально представлена архитектура ядра системы и представлены потоки выполнения и данных (раздел 3.6). В конце главы приводятся временные характеристики и особенности разработанной системы (раздел 3.7).

Четвертая глава диссертационной работы посвящена основным результатам моделирования процесса направленного развития. Глава состоит из двух разделов. Раздел 4.1 содержит описание основных этапов проведенных исследований. Раздел 4.2 представляет основные результаты каждого из этапов, описанных в разделе 4.1.

Заключение содержит краткое описание полученных в работе результатов и их возможные применения.

В конце работы представлена копия справки о внедрении и использовании системы в Институте химической физики им. Н.Н. Семенова РАН.

ГЛАВА 1: ОБЗОР ОБЛАСТИ ИССЛЕДОВАНИЙ

Первая глава диссертационной работы представляет область исследований и основные возникающие задачи. Раздел 1.1 представляет обзор теории сетей и основных рассматриваемых в её контексте вопросов. Раздел 1.2 посвящен анализу существующих программных решений для работы с сетями и проблемам их применения для рассматриваемой задачи. В разделе 1.5 представлена методология и методы исследований.

1.1. Обзор теории сетей и рассматриваемых задач

В повседневной жизни нас окружают различные сложные системы. Взаимодействие генов, протеинов и метаболитов является неотъемлемой частью живой клетки и жизни в целом; передача сигналов между нейронами лежит в основе функционирования мозга и нервной системы животных и человека; реальное и виртуальное взаимодействие между людьми составляет основу социума и социальных сетей соответственно ([4]). За всеми этими системами стоят сложные сетевые структуры, которые представляют взаимодействия между компонентами системы. Исследованием и математическим описанием сетевых структур и происходящих в них процессах занимается относительно молодая междисциплинарная *теории сетей*. Теория сетей использует теорию графов из математики, статистическую механику из физики, глубинный анализ и визуализацию данных из информатики, статистический вывод из статистики и социальную структуру из социологии. Ключевым открытием теории является тот факт, что несмотря на явные различия между сложными системами, структура и эволюция сетей, стоящих за этими системами, подчиняются схожим наборам фундаментальных законов и принципов. Данный факт позволяет применять один и тот же математический аппарат для исследования упомянутых систем ([4]).

Толчком к быстрому развитию этой теории послужили две основополагающие публикации – [5] и [6], а также, частично, развитие информационных технологий, позволивших получать и обрабатывать большие объёмы данных реальных систем. Также значительным событием в развитии послужило открытие свойства «мир тесен» (small-world networks, [7] [8] [9]) и масштабно-инвариантных сетей (scale-free networks, [10] [11]). Несмотря на относительную молодость, теория сетей имеет большое влияние на различные сферы науки и жизни – начиная от менеджмента ([12]), биологии (*network biology*, [13]), медицины и здравоохранения (*network*

medicine, network pharmacology, [14]), безопасности и борьбы с терроризмом (*network-centric warfare*, [15]) и заканчивая эпидемиологией ([16] [17]) и неврологией ([18]).

Теория сетей занимается созданием механизмов исследования реальных сетей и выявления закономерностей в строении и в происходящих процессах в целях создания предиктивных моделей. Такие модели позволяют заменить реальные сети соответствующими моделями и проводить исследования на них. Под термином «модель» понимается процесс, который на основе входных параметров и некоторого заложенного алгоритма строит сеть с заданными характеристиками. Самой простой моделью, предложенной для моделирования реальных сетей, является классическая модель случайных графов. Существует два определения данной модели:

1) Модель $G(N,m)$: N вершин соединяются m случайными ребрами. Данная модель была введена и использована Эрдёшом и Рейни [5].
2) Модель $G(N,p)$: Каждая пара из N вершин соединяется независимо от других с вероятностью p. Данная модель была введена Гильбертом [19].

В первом варианте фиксируется общее количество ребер в сети, тогда как второй фиксирует вероятность наличия связи между любыми двумя вершинами. Чаще используется второй вариант ввиду двух основных причин: (i) большинство стандартных характеристик считать на данной модели гораздо легче и (ii) в реальных сетях, а также в ходе динамических процессов в них, количество связей редко остается фиксированным. Эрдеш и Реньи провели глубокий анализ данной модели [5] [20] и дали описание структурных изменений, происходящих в сети при изменении вероятности связи. В частности, они выделили пять основных фаз изменения строения сети при изменении параметров модели и описали структурный фазовый переход при рождении гигантской связной компоненты. Изучение гигантской компоненты и её свойств при различных условиях вызывает широкий интерес научного сообщества ([21], [22], [23], [24]). Другим важным явлением при изучении строения сети при изменении параметров является понятие перколяции (просачивания) и перколяционного перехода, который аналогичен фазовому переходу второго рода, ввиду чего также активно изучается ([25] [26] [27]).

Однако большинство реальных сетей значительно отличаются от случайных графов. Так, Ватц и Строгац описали свойство «мир тесен» («маленький мир») и показали, что социальные сети, связность сети Интернет, вики-сайты и генные сети проявляют данное свойство [9]. Они предложили соответствующую модель для теоретического исследования данного типа сетей – модель Ватца-Строгаца, для которой характерна (i) малая длина кратчайшего пути в среднем, и (ii) большой коэффициент кластеризации ([28], [29]). Далее, Барабаши и Альберт открыли

масштабно-инвариантные сети, которые характеризуются наличием «хабов» и степенного распределения степеней вершин. Они предложили соответствующую модель и алгоритм построения данного типа сетей. Эмпирически было установлено, что сеть соавторства математических статей, сети межбанковских систем платежей, семантические сети, и сети протеиновых взаимодействий имеют подобную структуру ([11] [30] [31]). Масштабно-инвариантная топология также наблюдается в высокотемпературных сверхпроводниках ([32]). Аналитически было показано, что масштабно-инвариантные сети являются сетями «ультрамалого мира» (ultra-small world, [33]). Исследование спектрального распределения масштабно-инвариантных сетей показало, что, в отличии от случайных сетей, для которых характерно полукруговое Вигнеровское распределение собственных значений матрицы смежности, масштабно-инвариантные сети характеризуются треугольной формой распределения спектра ([34]).

Значительную роль в исследовании структуры сетей играют различные локальные и глобальные характеристики. Так, сети «мир тесен» характеризуются логарифмической (в среднем) пропорциональностью роста расстояния между двумя произвольными вершинами от числа узлов в сети, а также высоким коэффициентом кластеризации. Расстояние между двумя вершинами – локальная характеристика сети, коэффициент кластеризации – глобальная. Ещё одна глобальная характеристика – распределение степеней узлов – играет важную роль в определении масштабной инвариантности сетей. Не менее важной локальной характеристикой сетей являются так называемые *мотивы* ([35] [36] [37]), которые являются повторяющимися и статистически важными подграфами или шаблонами. Идея мотивов является существенной, потому что позволяет пролить свет на принципы структурного образования и функциональных возможностей системы [38]. Одними из наиболее известных типов мотивов являются циклы с обратной/опережающей связью (feed-back/feed-forward loops, [39] [40]), содержащие три вершины. В данной работе рассматриваются подобные мотивы (трёх-вершинные *клики* – полные подграфы, содержащие три вершины), с той лишь разницей, что сети являются ненаправленными. Статистический анализ распределения различных трёх-вершинных мотивов (триад) показывает наличие четырех супер-семейств, определяющих функциональные свойства содержащихся в каждом из них сетей ([41]).

Реальные системы являются динамическими и с течением времени развиваются и изменяются, адаптируясь под новые условия. Важным направлением теории сетей является моделирование и изучение данных процессов и выявление закономерностей в их динамике. Термин «динамика» и равнозначный термин «эволюция» имеет два основных смысла. Первый – дина-

мика построения сети. Здесь подразумевается последовательное построение сети посредством определенной закономерности. Динамика в данном смысле позволяет исследовать структуру рассматриваемой сети и те свойства, которыми наделяется сеть данной динамикой. Примером такой динамики является процесс построения масштабно-инвариантных сетей по принципу предпочтительного присоединения ([28]). Второй – динамические процессы на сетях, когда изучаются различные изменения рассматриваемой сети на временной оси при внешних воздействиях. Процесс эволюции – яркий пример данной динамики ([42]). В контексте данной работы термин «динамика» используется во втором смысле, а именно рассматривается динамический процесс направленного развития сети во времени при наличии топологических ограничений. Этот процесс представляет из себя последовательные модификации сети с целью улучшения необходимых свойств и, поэтому, называется также целевым проектированием. Данная динамика играет важную роль при исследовании сетей, поскольку позволяет получать/развивать сети с необходимыми характеристиками на статистической основе. Одним из значительных результатов применения данной динамики является понимание механизма разделения распределений мотивов по отдельным стабильным состояниям («*острова стабильности* в море всех распределений мотивов», [43] [44]).

Кроме упомянутых, активно исследуются различные аспекты сложных сетей, такие как вопросы устойчивости (resilience) и надежности сетей ([45]), явления синхронизации ([46] [47]), распространение возмущений ([48]), спектральные явления ([49]), коллективное поведение ([50]). Одним из наиболее важных и быстроразвивающихся направлений является изучение критических явлений в сложных сетях. Критические явления охватывают широкий спектр вопросов: структурные изменения в сетях; возникновение критических масштабно-инвариантных архитектур; различные явления, связанные с перколяцией; эпидемиологические пороги; фазовые переходы в кооперативных моделях на сетях; критические точки различных проблем оптимизации; переходы в коэволюционных парах (кооперативная модель и её сетевой субстрат); переходы между различными режимами в процессах, протекающих в сетях и многие другие. Многие из этих явлений тесно связаны и универсальны для различных моделей и, поэтому, описываются с помощью универсального феноменологического подхода [51]. Наиболее изученным критическим явлением является структурный фазовый переход при образовании гигантской связной компоненты в случайных графах. Кроме того, получены значительные результаты для различных явлений в сетях, таких как перколяция, конденсация ребер и треуголь-

ников, переходы локализации, распространение эпидемий, процессы реакции-диффузии, модель избирателя (обзор соответствующих результатов приведен в [51]).

В данной работе рассматривается новое проявление структурного фазового перехода в случайных сетях в ходе процесса направленного развития сети и дается анализ происходящих изменений в структуре сети, в форме спектральной плотности матриц смежности и различных характеристик сети. Получение данных результатов стало возможным благодаря специально разработанной программной системе, проведенных с её помощью многочисленных компьютерных экспериментов и анализу полученных данных.

1.2. Программное моделирование сетей и процессов в них

Основным подходом проведения различных исследований сетей и происходящих в них явлений является компьютерное моделирование. Экспериментальные результаты позволяют подтвердить или опровергнуть различные гипотезы относительно функционирования и динамики сетей, а также позволяют получать и исследовать новые раннее неизвестные явления. Программное моделирование сетей охватывает множество направлений:

1) Структурный анализ сетей.
2) Графическое представление сетей.
3) Стандартные алгоритмы на сетях.
4) Узко специализированное моделирование сетей в различных областях.

Каждое из этих направлений располагает различными программными инструментами, спроектированными для решения соответствующих задач. Кроме того, существуют системы общего моделирования различных технических расчетов. Далее представлены некоторые известные системы и описаны их основные преимущества и недостатки.

1.2.1. Системы анализа и визуализации сетей

Основная часть доступных программных систем, работающих с сетями, предназначены для их статического анализа и визуального представления. Основные из них представлены ниже.

1. Tulip (http://tulip.labri.fr/TulipDrupal) - платформа визуализации информации, предназначенная для анализа и визуализации реляционных данных. Основная цель - предоставить разработчику полную библиотеку, поддерживающую разработку интерактивных приложений визуализации информации для реляционных данных, которые могут быть адаптированы к конкретным проблемам.

2. Cytoscape (http://www.cytoscape.org/) – платформа для визуализации сетей молекулярных взаимодействий и биологических путей и интеграции этих сетей с аннотациями, генетической информацией и другими данными состояния. Изначально система была предназначена для биологических исследований, однако в настоящее время это общая платформа для комплексного сетевого анализа и визуализации.
3. Gephi (https://gephi.org/) – система анализа и исследования графов. Цель – помочь аналитикам данных делать гипотезы, интуитивно обнаруживать шаблоны, изолировать особенности структуры или ошибки во время поиска данных. Позиционируется как дополнительный инструмент для традиционной статистики, так как визуальное мышление с интерактивными интерфейсами позволяет облегчить рассуждения.

1.2.2. Системы моделирования процессов в сетях

Существуют специализированные системы, позволяющие моделировать различные процессы в специфичных сетях. Основные из них:

1. ProtNet [52]– инструмент для стохастических симуляций динамики сетей протеиновых взаимодействий.
2. ns (http://www.nsnam.org/), OPNET (http://www.opnet.com/), NetSim (http://tetcos.com/) – системы моделирования и эмуляции компьютерных сетей. Позволяют исследовать различные вопросы построения, функционирования, безопасности и производительности компьютерных сетей.
3. Neuron (http://www.neuron.yale.edu/neuron/), GENESIS (http://genesis-sim.org/) – системы для моделирования нейронов, нейронных сетей мозга и других нейробиологических систем на различных уровнях.

Сложностью использования данных систем для рассматриваемой задачи является специализация этих систем для решения конкретного круга задач из соответствующей предметной области.

1.2.3. Системы общего назначения для технических расчетов

Для моделирования сетей и процессов в сетях возможно применение систем общего назначения. Одной из самых распространенных таких систем является пакет Mathematica (https://www.wolfram.com/mathematica/) компании Wolfram Research. Данный пакет предоставляет большой набор доступных функций и, в частности, средства работы с сетями. Кроме дан-

ного пакета, распространёнными являются также системы численного расчета MATLAB (https://www.mathworks.com/products/matlab.html) и Octave (https://www.gnu.org/software/octave/). Подобные системы предоставляют пользователю специальный язык, позволяющий программировать необходимые действия. Однако для этого необходимо, в первую очередь, знакомство с соответствующим языком среды, что сходно с знакомством с некоторым языком программирования. Кроме того, изменения рассматриваемой динамики (например, смена рассматриваемого свойства) будет приводить к изменению всего кода, что обусловлено отсутствием абстракции в общем и, в частности, объектной ориентированности. Поэтому, такие системы обладают меньшей гибкостью по сравнению с низкоуровневыми языками программирования, применяемых для разработки приложений технических расчетов. Поэтому, с одной стороны данные системы позволяют моделировать любые процессы, что делает их общеприменимыми, но с другой стороны, требуют пошагового описания необходимых действий, что ставит использование подобных систем на один уровень с разработкой приложений на языках программирования общего назначения, но предоставляют меньше гибкости по сравнению с последними.

В контексте поставленной задачи представленные системы обладают одним из следующих недостатков:

1. Системы не позволяют рассматривать динамики на сетях – они предназначены для статического анализа конкретных сетей, что включает расчет основных локальных и глобальных характеристик сетей и их визуализацию при помощи различных алгоритмов.
2. Системы являются специализированными для моделирования конкретных типов сетей и специфичных процессов в них. Примером таких систем, представленных выше, являются системы моделирования сетей протеиновых взаимодействий и их развития, системы моделирования передачи данных в компьютерных сетях и другие подобные системы.
3. Системы предназначены для различных численных расчетов и предполагают как знание соответствующего языка этих систем, так и программирование соответствующих алгоритмов и действий для получения требуемых результатов.

Ввиду приведенных недостатков использование описанных систем не обеспечивает гибкого и расширяемого программного решения поставленной задачи. Поэтому, после проведенного анализа готовых систем, была спроектирована и реализована программная система, специализированная для моделирования процесса направленного развития сетей (описание системы представлено в главе 3 диссертационной работы). Работа с сетями и реализация стандартных

алгоритмов возложена на подключаемую готовую библиотеку представления сетей. Далее приведены некоторые распространенные библиотеки, используемые при решении задач с сетями.

1.2.4. Библиотеки представления и моделирования сетей

Далее представлены основные библиотеки, предоставляющие базовые функции для программного моделирования сетей.

1. Boost Graph Library (http://www.boost.org/doc/libs/1_64_0/libs/graph/doc/index.html) – составляющая часть известной библиотеки Boost языка программирования Си++, позволяющая хранить графы и сети в памяти различными способами и предоставляет реализацию большинства известных алгоритмов на графах и сетях.
2. GraphStream (http://graphstream-project.org/) – Java-библиотека для моделирования и анализа динамических графов. Предоставляет возможности создавать, импортировать, экспортировать, измерять, размещать и визуализировать графы и сети.
3. LEMON (Library for Efficient Modeling and Optimization in Networks, http://lemon.cs.elte.hu/trac/lemon) - библиотека шаблонов Си++, обеспечивающая эффективную реализацию общих структур данных и алгоритмов с акцентом на задачах комбинаторной оптимизации, связанных главным образом с графами и сетями.
4. NetworkX (https://networkx.github.io/) - программный пакет на языке Python для создания, манипулирования и изучения структуры, динамики и функций сложных сетей.
5. NetworKit (https://networkit.iti.kit.edu/) - это развивающийся набор инструментальных средств с открытым исходным кодом для крупномасштабного сетевого анализа. Цель - предоставить инструменты для анализа больших сетей в диапазоне размеров от тысяч до миллиардов ребер. Для этого в ней реализованы эффективные алгоритмы на графах, многие из которых параллельны и используют преимущества многоядерных архитектур. Предоставляет возможности для вычисления стандартных характеристик сетей, таких как распределение степеней, коэффициенты кластеризации и меры центральности. В этом отношении NetworKit сопоставим с такими пакетами, как NetworkX, хотя и с акцентом на параллелизм и масштабируемость.
6. PowerGraph (https://github.com/jegonzal/PowerGraph, [53]) - высокопроизводительная распределенная вычислительная среда, написанная на Си++ и предназначенная для работы с графами.

Данные библиотеки предоставляют базовую функциональность и предполагают их подключение и использование для решения конкретных задач и разработки соответствующих программ. Для разработанной системы была использована библиотека Boost Graph Library из-за следующих основных преимуществ:

1. Данная библиотека является частью библиотеки Boost – одной из самых известных и многофункциональных библиотек для языка Си++, благодаря чему активно тестируется и развивается широким сообществом программистов. Это обеспечивает высокое качество кода, гибкость предоставляемых интерфейсов, расширяемость и стабильность. Кроме того, библиотека тесно интегрирована с другими компонентами библиотеки Boost, что значительно упрощает процесс разработки приложений.
2. Библиотека поддерживает большинство известных платформ, благодаря чему обеспечивается кроссплатформенность разрабатываемых на её основе приложений.
3. Библиотека содержит эффективные реализации множества известных алгоритмов на графах и сетях.
4. Библиотека имеет версию, поддерживающую параллелизм и распределение вычислений – Parallel Boost Graph Library.

1.3. Методология и методы исследования

Методологической и теоретической основой диссертационной работы послужили труды ведущих специалистов в области теории сетей и её применения в различных областях фундаментальной науки, а также специалистов в области программного моделирования сетевых структур и динамических процессов в них.

При проектировании и реализации необходимых программных средств произведен анализ находящихся в свободном доступе программных решений, их преимуществ и недостатков. Для выбора технологий моделирования и аппаратной поддержки применен метод сравнительного анализа.

Для достижения необходимых характеристик системы, применен экспериментальный метод с обратной связью на основе сравнительного анализа полученных характеристик с требуемыми. Цикл метода включает четыре основных шага: разработка, тестирование, анализ результатов, выявление и удаление недостатков.

Экспериментальный метод применен для получения результатов по целевому проектированию сетей, а для анализа полученных данных применен сравнительный анализ полученных и существующих результатов.

ГЛАВА 2: ТЕОРЕТИЧЕСКАЯ И ТЕХНОЛОГИЧЕСКАЯ ОСНОВА

Глава посвящена теоретическим и практическим результатам, использованным в диссертационной работе. Раздел 2.1 представляет понятие случайных графов, их математическое представление, описываются основные свойства и характеристики сетей, методы двумерной визуализации сетей, алгоритмы обнаружения сообществ, а также спектральные свойства матриц в целом и матриц смежности сетей в частности. Раздел 2.2 представляет понятие семплирования и алгоритм Метрополиса-Гастингса, лежащий в основе рассматриваемого процесса направленного развития сетей. Раздел 2.3 описывает модель процесса направленного развития сетей. Раздел 2.4 представляет технологическую основу программного моделирования процесса - представлены основные методы и технологии для параллельных и распределенных вычислений.

2.1. Случайные графы и сложные сети

В математике под термином случайный граф подразумевается вероятностное распределение графов. Случайные графы могут быть описаны просто распределением вероятности или же с помощью случайного процесса, который генерирует их. Теория случайных графов находится на пересечении теории графов и теории вероятностей. Математическим аспектом применения случайных графов является выяснение свойств типичных графов. Практическое применение случайных графов охватывает все те области, где необходимо моделирование сложных сетей. Известно большое множество моделей случайных графов, что отражает большое разнообразие сложных сетей в различных областях. В математическом контексте термин случайный граф почти всегда означает модель случайных графов Эрдёша–Реньи (ER). В иных контекстах термин случайный граф может обозначать любую модель графов ([54]).

2.1.1. Модель случайных графов Эрдёша – Реньи

Построение случайного графа начинается с пустого графа с n изолированными вершинами. Далее, между вершинами последовательно случайным образом добавляются рёбра. Основной целью изучения данных графов является выявление того этапа, на котором появляется определённое свойство. Для получения различных распределений вероятностей на графе применяются различные модели случайных графов. Наиболее распространенным и изучаемым является распределение Гильберта с обозначением $G(n, p)$. В данном случае любое возможное ребро добавляется независимо от других с вероятностью $0 < p < 1$. Вероятность того, что слу-

чайный граф будет иметь в точности k рёбер равна $p^k(1-p)^{N-k}$, где $N=\binom{n}{2}$. Близкая к модели Гильберта модель Эрдёша–Реньи, обозначаемая $G(n,M)$, определяется на множестве всех графов с n вершинами и M ребрами и даёт каждому из них одинаковую вероятность. При $0\le M\le N$, $G(n,p)$ будет содержать $\binom{N}{M}$ элементов и вероятность выпадения каждого элемента составляет $\binom{N}{M}^{-1}$. Если рассматривать случайный процесс на графе $\tilde{G}_n$, который начинается с n вершин без рёбер и на каждом шагу добавляется новое, выбранное равномерно из множества несвязных пар вершин, ребро, то данную модель можно рассматривать как снимок некоторого момента (M) этого процесса ([54]).

Другая модель, являющаяся обобщением модели Гильберта случайного графа – это модель случайного скалярного произведения. В данной модели с каждой вершиной графа связывается вещественный вектор. Вероятность наличия ребра (u,v) между любыми двумя вершинами u и v является определенной функцией от скалярного произведения $u\cdot v$ соответствующих этим вершинам векторов ([54]).

Матрицы вероятности сети моделируют случайные графы при помощи вероятности рёбер $p_{i,j}$ таким образом, что ребро $e_{i,j}$ существует в определенный период времени. Данную модель можно обобщить на взвешенные и не взвешенные, ориентированные и неориентированные, статические и динамические графы.

Для $M \square pN$, где N является максимальным возможным числом рёбер, чаще всего принято использовать модели $G(n,M)$ и $G(n,p)$, которые почти всегда взаимозаменяемы ([55]).

Если рассматривать модель случайных графов, то любая функция, заданная на графах, превращается в случайную величину. Задачей изучения подобной модели является определение, или, по крайней мере, оценка вероятности появления некоторого свойства.

Эрдёш и Реньи в своей первой статье, посвященной случайным графам, использовали понятие случайного графа как N помеченных вершин, которые соединены n ребрами. Ребра выбираются случайным образом из множества $\frac{N(N-1)}{2}$ всех возможных ребер ([5]). Общее количество графов, имеющих N вершин и n ребер, составляет $C^{n}_{\frac{N(N-1)}{2}}$. Множество этих графов образует вероятностное пространство и каждая реализация имеет равную вероятность.

Другим определением случайного графа является биноминальная модель. В этом случае, каждые две из имеющихся N вершин соединяются с вероятностью p. В результате, полученное количество ребер является случайной величиной, обладающей ожидаемым значением $N \to \infty$. Если G_0 представляет граф с вершинами $v_1, v_2, ..., v_N$ и n ребрами, то вероятность его получения при помощи рассматриваемого процесса составляет $P(G_0) = p^n (1-p)^{\frac{N(N-1)}{2}-n}$ ([56]).

Основной задачей теории случайных графов является изучение вероятностного пространства графов с $N \to \infty$ вершинами. Множество свойств подобных случайных графов может быть получено при помощи случайного анализа. Согласно определению Эрдёша и Реньи, свойство Q имеет место для каждого графа, если вероятность выполнения Q равна 1 при $N \to \infty$. Некоторые из рассмотренных Эрдёшем и Реньи вопросов имеют непосредственное отношение к сложным сетям. Примерами таких вопросов являются следующие: Является ли стандартный граф связным? Содержится ли в графе цикл длины три? Каким образом диаметр графа зависит от его размеров? ([56])

Процесс построения случайного графа часто называют *эволюцией*: процесс начинается с N изолированных вершин и последовательно развивается посредством добавления новых случайных ребер [56]. В ходе процесса, результирующие графы соответствуют все большим значениям вероятности p, и, в предельном случае $p = 1$, получается полный граф, имеющий максимально возможное количество ребер $n = \frac{N(N-1)}{2}$.

Одна из основных идей теории случайных графов заключается в том, чтобы определить значение вероятности p, при котором проявляется некоторое свойство. Значительным открытием Эрдёша и Реньи является то, что многие значимые свойства случайных графов начинают проявляться достаточно внезапно ([20]). Это означает, что при фиксированной вероятности, или почти каждый граф обладает некоторым свойством Q, или же почти ни один граф этим свойством не обладает. При этом переход из одного состояния в другое происходит очень резко. Для множества свойств существует критическая вероятность $p_c(N)$. Когда $p(N)$ растет медленнее, чем $p_c(N)$ при $N \to \infty$, то практически ни один граф не обладает свойством Q. Когда $p(N)$ растет быстрее, чем $p_c(N)$, то практически любой граф обладает свойством Q ([56]). Следовательно, вероятность того, что граф с N вершинами и функцией распределения ребер $p = p(N)$ обладает заданным свойством Q, определяется следующим образом:

$$\lim_{N\to\infty} P_{N,p}(Q) = \begin{cases} 0, если \dfrac{p(n)}{p_c(n)} \to 0 \\ 1, если \dfrac{p(n)}{p_c(n)} \to \infty \end{cases} \quad (1)$$

Наконец, в теории случайных графов вероятность определяется как функция от размера системы: p является дробью от максимально возможного количества рёбер $\frac{N(N-1)}{2}$. Графы большего размера с одинаковым p будут содержать больше ребер, и, в конце концов, свойства, подобные появлению циклов, скорее проявятся в них, нежели в графах меньшего размера. Это означает, что нет единственного (не зависящего от N) допуска для многих свойств случайных графов и появляется необходимость определения допуска как функции от размера системы, при $p_c(N\to\infty)\to 0$. С другой стороны, верно утверждение, что средняя степень графа обладает критическим значением, не зависящим от размера системы ([56]).

2.1.2. Основные характеристики сетей

Свойства случайных графов играют важную роль в ходе исследования поведения графов во время различных случайных процессов. Рассмотрим некоторые из них, примененных в ходе исследований в рамках данной диссертационной работы.

Распределение степеней. Степень k_i вершины i в случайном графе с N вершинами и вероятностью связности p следует биномиальному распределению с параметрами $N-1$ и p:

$$P(k_i = k) = C_{N-1}^{k} \cdot p^k \cdot (1-p)^{N-1-k} \quad (2)$$

Данная вероятность представляет число способов, которыми можно провести k ребер из фиксированной вершины: так как вероятность существования одного ребра составляет p, то вероятность существования k ребер составит p^k; вероятность отсутствия всех остальных ребер из данной вершины составляет $(1-p)^{N-1-k}$; существует C_{N-1}^{k} способов выбора k вершин, с которыми будет соединена вершина. Для нахождения распределения степеней графа, необходимо рассмотреть количество вершин, обладающих степенью k (обозначим это число через X_k). Необходимо определить вероятность того, что X_k принимает заданное значение $P(X_k = r)$.

Согласно формуле для $P(k_i = k)$, ожидаемое количество вершин со степенью k составит $E(X_k) = N \cdot P(k_i = k) = \lambda_k$, где $\lambda_k = N \cdot C_{N-1}^{k} \cdot p^k \cdot (1-p)^{N-1-k}$. С достаточно хорошим приближением для распределения степеней случайного графа подходит биномиальное распределение

$P(k) = C_{N-1}^{k} \cdot p^{k} \cdot (1-p)^{N-1-k}$, которое для больших N может быть заменено на $P(X_k = r) = \frac{\lambda_2}{r!} e^{-\lambda_2}$ ([57]).

Распределение степеней, как уже было представлено в главе 1, играет важную роль при классификации сетей, что, в свою очередь, определяет функциональные характеристики сетей.

Связность и диаметр. Максимальное расстояние всеми парами вершин графа называется диаметром. Для несвязного графа (например, если граф состоит из нескольких несвязных кластеров) диаметр определяется либо как бесконечность, либо как максимальный из диаметров составляющих его кластеров. Для случайных графов при малых p свойственен малый диаметр. Причина этого заключается в том, что случайный граф кажется расширяющимся: с большой вероятностью количество вершин с расстоянием l от фиксированной вершины пропорционально величине $\ln(N)/\ln(\langle k \rangle)$, где $\langle k \rangle$ – средняя степень вершины, то есть, оно логарифмически зависит только от размера графа. Многие ученные исследовали диаметр случайных графов ([58]). Общий вывод следующий – для большинства значений p, практически все графы имеют одинаковый диаметр. Это значит, что, если рассматривать все графы с N вершинами и вероятностью связности p, их диаметры могут лишь незначительно отличаться, обычно находясь в малой окрестности значения

$$d = \frac{\ln(N)}{\ln(p \cdot N)} = \frac{\ln(N)}{\ln(\langle k \rangle)} \qquad (3)$$

Кластерный коэффициент. Если рассматривать некоторую вершину случайного графа и ее ближние соседние вершины, то вероятность существования связи между двумя соседними вершинами, равна вероятности того, что две случайно выбранных вершины соединены. Как следствие, кластерный коэффициент случайного графа составляет:

$$C_{rand} = p = \frac{\langle k \rangle}{N} \qquad (4)$$

Исследование отношения $C_{rand}/\langle k \rangle$ как функции от N для случайных графов разных размеров показало, что реальные сети не следуют предсказанию для случайных графов. Отношение не увеличивается как N^{-1}; вместо этого оно оказывается независимым от N. Это свойство характерно для больших упорядоченных сетей, коэффициент кластеризации которых зависит только от числа координированности сети, а не от его размера ([9]).

Локальный кластерный коэффициент определяет плотность связей вокруг каждой вершины сети, тогда как глобальный коэффициент кластеризации является характеристикой плотности связей в сети в целом.

Среднее расстояние между вершинами. Расстоянием между двумя вершинами графа называется число рёбер в кратчайшем пути между этими вершинами (также называемым геодезической графа). Расстояние в графе принято также называть геодезическим расстоянием. Между двумя вершинами может существовать несколько кратчайших путей. Если между двумя вершинами нет пути – если они принадлежат различным компонентам связности – то расстояние принято считать бесконечным [59].

Усредненное по всем парам вершин в графе расстояние называется средним расстоянием между вершинами графа. Данная величина, на ряду с диаметром графа, характеризует его компактность.

2.1.3. Силовые алгоритмы визуализации сетей

Для построения изображений неориентрованных графов широко используются силовые алгоритмы. Такие алгоритмы относят к наиболее гибким алгоритмам, основой которых являются физические аналогии. Их популярность обусловлена следующими причинами:

- Применимость для графов любого вида.
- Адаптация к специфическим требованиям.
- Интуитивность.
- Простота реализации.
- Эффективность применения для графов с количеством вершин порядка ста пятидесяти.

Изображениям, которые получены с помощью силовых алгоритмов, характерны следующие основные свойства:

- малое пересечение ребер;
- симметричность;
- эстетичность.

В основу силовых алгоритмов входят две компоненты ([60]):

1. модель, которая описывает систему физических объектов и их взаимосвязи;
2. алгоритм, который находит состояние равновесия для рассматриваемой системы.

Описание модели связано с понятием о “хорошести” изображения в каждом отдельном случае. Для модели определяется целевая функция, которая и описывает понятие “хорошести”, а силовой алгоритм нацелен на оптимизацию целевой функции. После описания системы объек-

там разрешено движение, обусловленное действующими на них силами. Это движение постепенно приведет систему в состояние равновесия (в этом состоянии все силы будут уравновешивать друг друга). Состоянию равновесия будет соответствовать такое изображение, которое приблизительно удовлетворяет вышеназванным критериям. Рассматриваемую модель можно выразить двумя основными способами:

- в терминах потенциальной энергии
- в терминах сил, которые действует на физические объекты.

Методы, в основе которых лежит понятие энергии, также состоят двух компонент: модели энергии и ищущего состояние с минимальной энергией алгоритма. Силовые методы основаны на модели сил и алгоритме, который ищет положение, в котором результирующая сила, действующая на каждый объект, равна нулю. Положение равновесия, в которое постепенно придет система, соответствует локальному минимуму энергии, в силу того, что сила есть отрицательный градиент энергии.

Свое первоначальное применение силовые алгоритмы размещения получили в задаче об автоматическом построении топологии интегральных схем. В 1984 году Питер Идес (P. Eades) предложил алгоритм размещения для изображения графов, которые имеют максимум 30 вершин ([61]). Для графа $G(V,E)$ в алгоритме Идеса вершинам V ставились в соответствие стальные шары, а ребрам E – пружины, соединяющие эти шары. Рассматривалась задача о нахождении наиболее симметрического размещения, при котором все вершины, связанные ребром, должны были располагаться на приблизительно равном расстоянии друг от друга. Это расстояние называлось «естественной» длиной ребра. В силу того, что в большинстве случаев невозможно изобразить весь граф с прямыми ребрами одинаковой длины, неизбежным является некоторое искажение естественной длины. Задача, о существовании для произвольного графа прямолинейного размещения с ребрами равной длины является NP-полной ([62]).

Пусть $G=(V,E)$ связный неориентированный граф. Обозначим $P=(p_v), v\in V$ – вектор позиций вершин на плоскости. Вершине $v\in V$ соответствует координата $p_v=(x_v,y_v)$. Первоначально осуществляется размещение вершин-шаров в некоторое случайное начальное состояние. Расстояние между произвольными вершинами $u,v\in V$ определяется как Евклидово расстояние и обозначается $\|p_v-p_u\|$.

Далее,

$$\overline{p_u p_v}=\frac{p_v-p_u}{\|p_v-p_u\|} \qquad (5)$$

обозначает единичный вектор, направленный от p_v к p_u.

Для того, чтобы смежные вершины располагались примерно на одинаковом расстоянии, вводится сила пружины, действующая между соседними вершинами ([61]):

$$f_{spring}(p_u, p_v) = c_{spring} \log \frac{\|p_u - p_v\|}{l} \overrightarrow{p_v p_u} \tag{6}$$

где c_{spring} – это параметр управления силы пружины, l - «естественная» длина пружины.

«Сила пружины», введенная Идесом, не соответствует классическому закону Гука. Эксперименты показали, что сила, испытываемая пружиной, слишком сильна, когда вершины находятся далеко друг от друга, по этой причине была введена искусственная «логарифмическая» пружина, действующая слабее на более удаленные вершины. Логарифм обеспечивает следующее свойство: если реальная длина пружины меньше естественной длины, пружина растягивается, а если реальная длина пружины больше естественной длины, пружина сжимается.

Для получения наиболее симметричных изображений применяются силы отталкивания. Определяются такие силы для произвольной пары несмежных вершин $u, v \in V$ следующим образом ([61]):

$$f_{rep}(p_u, p_v) = \frac{c_{rep}}{\|p_v - p_u\|^2} \overrightarrow{p_u p_v} \tag{7}$$

где c_{rep} является постоянной. Для получения конфигурации равновесия для заданной системы, вершины сдвигаются в каждый момент времени t в соответствии с вектором результирующих сил $F_v(t)$. Для произвольной вершины $v \in V$ вектор $F_v(t)$ определяется как сумма всех действующих на вершину v пружинных сил и сил отталкивания. После вычисления $F_v(t)$ для всех $v \in V$ каждая вершина сдвигается по этому вектору на величину $F_v(t)$. Константа используется для предотвращения излишнего перемещения вершин вследствие синхронных изменений позиций. Производя итеративное вычисление сил, которые действуют на каждую вершину и изменяя соответственно позиции этих вершин, система переводится в стабильное состояние, в котором невозможны дальнейшие локальные улучшения. Процедура поиска положения равновесия системы имеет следующий вид.

1: **PROCEDURE** SPRING_EMBEDDER(G)

2: **FOR** $t = 1..M$

3: **DO**

4: **FOR** $v \in V$

5: **DO**

6: $F_v(t) = \sum_{u:(u,v)\in E} f_{rep}(p_u, p_v) + \sum_{u:(u,v)\in E} f_{spring}(p_u, p_v)$

7: **DONE**

8: **FOR** $v \in V$

9: **DO**

10: $p_v = p_v + \delta F_v(t)$

11: **DONE**

Алгоритм 1. Алгоритм Spring Embedder

Алгоритм Spring Embedder (Eades 1984, [61]***)***

Вход: На вход алгоритм получает связный неориентированный граф $G = (V, E)$ и некоторое начальное (часто используется случайное) размещение его вершин $p = (p_v)$

Результат: Размещение равновесного положения системы.

Метод: Алгоритм 1 представляет последовательность действий, выполняемых для получения необходимого размещения.

Для большинства графов подходящими являются значения $c_{spring} = 2, c_{rep} = 1, l = 1, \delta = 0,1.$ Переход в состояние с минимальной энергией в основном происходит при $M = 100$.

Алгоритм Фрухтермана и Рейнгольда (1991, [63]***)***

В алгоритме, в 1991 году предложенном Фрухтерманом и Рейнгольдом, к представленным двум критериям добавлено требование о «равномерном распределении вершин». Также немного модифицированы определения сил, действующих на вершины. В основном модификация сил была направлена на улучшение времени работы алгоритма. Результаты, полученные данным алгоритмом, достаточно схожи с результатами алгоритма Идеса. С целью удовлетворения критерия равномерного распределения вершин расчет идеальной длины ребра стал производиться при помощи площади размещения и количества вершин:

$$l = \sqrt{\frac{area}{numberOfVertices}} \qquad (8)$$

Подсчет результирующих сил производится по следующим правилам ([60]).

– Силы отталкивания действуют между всеми парами вершин и вычисляются следующей формулой

$$f_{rep}(p_u, p_v) = \frac{l^2}{\|p_u - p_v\|} \overrightarrow{p_u p_v} \tag{9}$$

для всех $(u,v) \in V$.

– Силы притяжения действуют только между смежными вершинами и рассчитываются посредством следующей формулы:

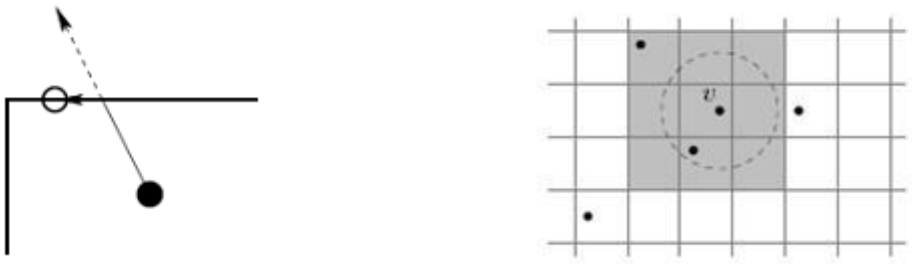
$$f_{attr}(p_u, p_v) = \frac{\|p_u - p_v\|^2}{l} \overrightarrow{p_v p_u} \tag{10}$$

– Результирующая сила пружины вычисляется как сумма сил отталкивания и сил притяжения:

$$F_{spring}(p_u, p_v) = f_{attr}(p_u, p_v) + f_{rep}(p_u, p_v) \tag{11}$$

В процессе поиска равновесия системы постоянный множитель δ заменен на функцию $\delta(t)$, которая зависит от времени. Вводимая функция $\delta(t)$ является убывающей для избежания излишних изменений позиций вершин, в особенности на тех этапах работы алгоритма, когда текущее размещение приближается к состоянию равновесия. Вводится новое понятие «температуры». В начале работы алгоритма задается начальное значение температуры, например, как десятая часть ширины области размещения ([64]). Далее в процессе работы алгоритма температура убывает до нуля. С помощью температуры перемещения вершин происходят таким образом, что по мере улучшения размещения, уменьшаются изменения позиций вершин. Данная идея заимствована у методов имитации отжига. Для этого алгоритма характерной особенностью также является обрезка. Под обрезкой подразумевается модификация расположений вершин, для гарантии размещения вершин внутри прямоугольной области экрана и/или бумаги. Если результирующий вектор смещения выводит вершину за границу области, то соответствующая координата вектора смещений обрезается, как это показано на Рис. 1(a). Другая особенность данного алгоритма связана с изменением точности вычислений. С этой целью алгоритм был разработан сеточный вариант алгоритма. При вычислении суммы сил отталкивания несущественные вершины выбрасываются из этой суммы при помощи применения сеточной техники. Несущественными считаются отдаленные вершины, в силу того, что отталкивание таких вершин не имеет сильного влияния на вектор смещения. При вычислении сил отталкивания учитываются только те вершины, которые расположены в узлах сетки близких к узлу, в

котором находится вершина v и только если их расстояние меньше заданного порога. (Рис.



1(б)).

Применение такого метода для разреженных графов с равномерно распределенными вершинами позволяет получить оценку $O(n)$ времени при вычислении сил отталкивания, но такое улучшение получается за счет потери качества размещения.

2.1.4. Алгоритмы обнаружения сообществ. Алгоритм Louvain

В теории сложных сетей принято говорить, что сеть имеет общинную структуру, если узлы сети могут быть легко сгруппированы в (возможно, перекрывающиеся) наборы узлов, так что каждый набор узлов плотно связан внутри. В частном случае поиска непересекающихся сообществ это означает, что сеть естественно делится на группы узлов с плотными связями внутри групп и более редкими связями между группами. Но также допускается пересечение сообществ. Более общее определение основано на принципе, что пары узлов, скорее всего, будут связаны, если они оба являются членами одного и того же сообщества (общин), и менее вероятно, будут связаны, если они не находятся в одном сообществе.

Поиск сообществ в произвольной сети может быть вычислительно сложной задачей. Коли-

Рисунок 1. Особенности алгоритма Фрухтермана-Рейнгольда. (а) Обрезка области размещения. (б) локальное вычисление сил отталкивания.

чество сообществ, если таковые имеются, внутри сети, как правило, неизвестно, и общины часто имеют неодинаковые размеры и/или плотность. Несмотря на эти трудности, были разработаны и применяются различные методы поиска сообществ, имеющие различные степени успеха ([65]). Основными методами являются: метод минимального разреза, иерархическая кластеризация, алгоритм Гирвана-Ньюмана, максимизация модулярности, методы статистического вывода и методы, основанные на кликах.

Одним из алгоритмов поиска сообществ, основанных на максимизации модулярности, является алгоритм Louvain ([66]). Метод Louvain предназначен для обнаружения и извлечения сообществ из больших сетей. Авторы – В. Блондель, Ж.-Л. Гийом, Р. Ламбиот и Э. Лефевр. Этот метод является жадным алгоритмом, который имеет временную сложность $O(n\log n)$.

Вдохновением для этого метода обнаружения сообщества является оптимизация модульности по мере продвижения алгоритма. Модульность - это масштабное значение от -1 до 1, которое измеряет плотность ребер внутри сообществ к ребрам вне сообществ. Оптимизация этого значения теоретически приводит к наилучшей возможной группировке узлов данной сети, однако прохождение всех возможных итераций узлов в группах нецелесообразно, поэтому используются эвристические алгоритмы. В методе Louvain обнаружения сообщества первые небольшие сообщества обнаруживаются путем оптимизации модульности локально на всех узлах, затем каждое небольшое сообщество группируется в один узел, и первый шаг повторяется.

Алгоритм Louvain

Оптимизируемое значение - это модульность, определяемая как значение от -1 до 1, которое измеряет плотность связей внутри сообществ по сравнению с связями между сообществами ([66]). Для взвешенного графа модульность определяется как:

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j), \qquad (12)$$

где

A_{ij} представляет вес ребер между узлами i и j;

k_i и k_j - это сумма весов ребер, связанных с узлами i и j соответственно;

m - сумма всех весов ребер в графе;

c_i и c_j - сообщества узлов; а также

δ - простая дельта-функция.

Для того чтобы эффективно максимизировать это значение, метод Louvain имеет две фазы, которые итеративно повторяются.

Во-первых, каждому узлу сети назначается своё сообщество. Затем для каждого узла i измеряется изменение модульности при удалении i из своего сообщества и перемещения его в сообщество каждого соседа j узла i. Это значение легко вычисляется по следующей формуле:

$$\Delta Q = \left[\frac{\Sigma_{in} + 2k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \quad (13)$$

Где Σ_{in} - сумма весов всех ребер внутри сообщества, в которое перемещается узел i, Σ_{tot} - это сумма весов всех ребер к узлам сообщества, k_i - это взвешенная степень узла i, $k_{i,in}$ - это сумма весов ребер между узлом i и другими узлами в сообществе и m - сумма весов всех ребер в сети. Затем, после того, как это значение будет рассчитано для всех сообществ, с которыми связан узел i, узел i будет перемещён в то сообщество, которое привело к наибольшему увеличению модульности. Если увеличение невозможно, i остается в исходном сообществе. Этот процесс применяется многократно и последовательно ко всем узлам до тех пор, пока не произойдет увеличение модульности. Как только этот локальный максимум модульности будет достигнут, первая фаза закончится.

На второй фазе алгоритма он группирует все узлы в одних и тех же сообществах и строит новую сеть, где узлами являются сообщества из предыдущего этапа. Любые связи между узлами одного и того же сообщества в настоящее время представлены петлями на узле нового сообщества, а связи из нескольких узлов в одном сообществе с узлом в другом сообществе представлены взвешенными ребрами между сообществами. После создания новых сетей вторая фаза завершается, и первый этап может быть повторно применен к новой сети.

2.1.5. Спектральные свойства матриц смежности

Ненулевой вектор x называется собственным вектором квадратной матрицы A, если для некоторого скаляра λ имеет место $Ax = \lambda x$. Собственным значением квадратной матрицы A называется такое число λ, для которого существует собственный вектор, то есть уравнение $Ax = \lambda x$ имеет ненулевое решение x. Проще говоря, собственный вектор — любой ненулевой вектор x, который преобразуется матрицей A в коллинеарный вектор λx, а соответствующий скаляр λ называется собственным значением матрицы.

Одним из основных способов представления графа является его квадратная матрица смежности, следовательно, весь математический аппарат анализа собственных значений и собственных векторов применим к графам и позволяет рассматривать их как объекты линейной алгебры.

Матрица смежности неориентированного графа является симметричной, а значит обладает действительными собственными значениями и ортогональным базисом из собственных векторов. Набор её собственных значений называется *спектром* графа, и является основным пред-

метом изучения спектральной теории графов. Матрица смежности графа зависит от нумерации его вершин, однако его спектр является *инвариантом* графа и, поэтому, играет важную роль при проверке изоморфизма графов, а также в задачах компьютерной химии ([67]).

Закон полуокружностей Вигнера

Пусть A есть вещественная симметричная матрица большого порядка N с случайными элементами a_{ij}, которые для $i \le j$ равномерно распределены с равными плотностями, равными вторыми моментами m^2, и моменты порядка n ограничены константами B_n, независимо от i, j и N. Далее, пусть $S = S_{\alpha,\beta}(a, N)$ – число собственных значений A, лежащих в интервале $(\alpha N^{\frac{1}{2}}, \beta N^{\frac{1}{2}})$ для вещественных $\alpha < \beta$. Тогда

$$\lim_{N\to\infty} \frac{E(S)}{N} = \frac{1}{2\pi m^2} \int_{\alpha}^{\beta} \sqrt{4m^2 - x^2}\, dx \qquad (14)$$

([68] [69]). Данный закон первым получил Вигнер (1955) для некоторых специальных классов случайных матриц, рассматриваемых в области квантовой механики.

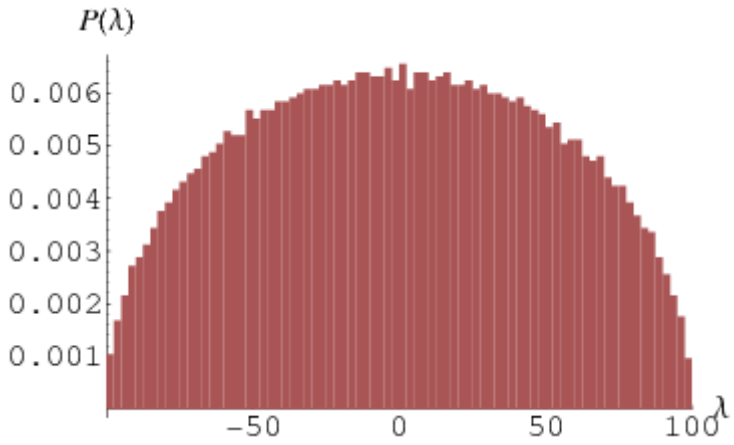


Рисунок 2. Распределение собственных значений случайной симметричной матрицы размером 5000 со значениями, полученными из стандартного нормального распределения

Стоит отметить, что большие вещественные матрицы со значениями из равномерного распределения тоже следуют закону полуокружностей с одним исключением – существует в точности одно большое собственное значение, оторванное от общей массы значений. Следовательно, именно такое распределение имеют матрицы смежности сетей Эрдёша-Реньи.

2.2. Семплирование. Алгоритм Метрополиса - Гастингса

В математической статистике процесс семплирования представляет из себя метод манипуляции с начальной выборкой, позволяющий выполнить структурно-параметрическую иденти-

фикацию наилучшей статистической модели случайного процесса. Научная новизна семплирования заключается в том, что оно является эффективным приемом для логичного смыслового увязывания статистических свойств выборки и цели моделирования ([70]).

Алгоритм Метрополиса-Гастингса является одним из методов семплирования и используется, в основном, для сложных функций распределения. Он отчасти схож с алгоритмом выборки с отклонением, однако здесь используется изменяемая со временем вспомогательная функция распределения. Алгоритм был впервые опубликован Н. Метрополисом в 1953 году ([71]), и затем был обобщён К. Гастингсом в 1970 году ([72]). Другой известный метод семплирования – семплирование по Гиббсу – является частным случаем данного алгоритма, но более популярен благодаря простоте и скорости, хотя и реже применим.

Алгоритм Метрополиса - Гастингса позволяет проводить семплирование любой функции распределения. В основе работы алгоритма лежит создание цепи Маркова – на каждом шаге алгоритма новое выбираемое значение x^{t+1} зависит только от значения x^t на предыдущем шаге. Алгоритм использует некоторую вспомогательную функцию распределения $Q(x'|x^t)$, зависящую от x^t, для которой генерировать выборку просто (распространенный пример - нормальное распределение). На очередном шаге для этой функции генерируется случайное значение x'. Затем с вероятностью

$$u = \frac{P(x')Q(x^t \mid x')}{P(x^t)Q(x' \mid x^t)} \tag{15}$$

(или с вероятностью 1, если $u > 1$), полученное значение становится новым: $x^{t+1} = x'$, а иначе оставляется старое: $x^{t+1} = x^t$. Ясно, что предложенная функция генерирует новое значение в зависимости от значения на предыдущем шаге ([70]).

Вариант алгоритма, изначально предложенный Метрополисом, требовал симметричности вспомогательной функции: $Q(x', x^t) = Q(x^t, x')$, однако дальнейшее обобщение Гастингса позволило избавиться от этого ограничения.

Пусть уже выбрано случайное значение на шаге t – x^t. Выбор следующего значения производится посредством генерации промежуточного значение x' для функции $Q(x'|x^t)$; далее вычисляется произведение $a = a_1 a_2$, где

$$a_1 = \frac{P(x')}{P(x^t)} \tag{16}$$

представляет отношение вероятностей между полученным промежуточным значением и предыдущим, и

$$a_2 = \frac{Q(x^t \mid x')}{Q(x' \mid x^t)} \tag{17}$$

это отношение вероятностей пойти из x' в x^t или обратно. Очевидно, что множитель a_2 в случае симметричности Q равен единице. Правило выбора случайного значения для нового шага имеет следующий вид:

Если $a \geq 1 : x^{t+1} = x'$

Если $a < 1 : x^{t+1} = \begin{cases} x' \text{ с вероятностью } a \\ x^t \text{ с вероятностью } 1-a \end{cases}$

Стартуя из некоторого случайно выбранного значения x^0, алгоритм сначала прогоняется на некотором количестве шагов «вхолостую», что позволяет «забыть» начальное значение. Алгоритм лучше всего работает в случае, когда вспомогательная функция достаточно близка по форме к целевой функции P ([70]). Однако априори добиться этого зачастую невозможно. В качестве решения для этой проблемы применяется подготовительная стадия работы алгоритма, которая позволяет должным образом настроить вспомогательную функцию.

2.3. Модель процесса направленного развития сетей

Прежде чем приступить к описанию модели, необходимо сделать следующее важное замечание: в научной литературе термины «сеть» и «граф» используется взаимозаменяемо. Наука о сетях использует термины «сеть», «узел» и «связь», тогда как теория графов использует, соответственно, термины «граф», «вершина» и «ребро». Существует небольшое различие между этими двумя терминологиями. Первая обычно используется для обозначения реальных систем – всемирная паутина, социальные сети, метаболические сети. Вторая терминология, в противоположность, используется для обозначения математического представления соответствующих сетей – веб-граф, социальный граф и метаболический граф, соответственно. Однако, это различие делается редко и обе терминологии в основном используются как синонимы ([4]). Поэтому далее по тексту работы эти термины будут применятся равнозначно.

Рассматривается классическая модель случайных графов Эрдёша-Реньи ([5]). В разделе 1.1 было дано описание данной модели: N идентичных вершин соединяются ненаправленными ребрами, причем каждое ребро добавляется с независимой вероятностью p. Модель случайных графов Эрдёша-Реньи задается с помощью пары $G = (V, E)$, где $V(G) = \{x_1, ..., x_N\}$ – непустое конечное множество вершин и $E(G)$ – конечное множество ребер. Модель обладает

симметричной матрицей смежности A с элементами a_{ij}, такими, что $a_{ij}=1$, если $(x_i, x_j) \in E(G)$ и $a_{ij}=0$ в противном случае.

Предположим задана некоторая сеть G. Обозначим через $T(G)$ количество циклов длины три в G. Далее, будем рассматривать процесс последовательных изменений сети с целью увеличения $T(G)$. Данный процесс представляет из себя последовательность трансформаций начального графа G – $\{G=G_0, G_1, G_2, ..., G_k\}, k \geq 1$. Каждый последующий граф получается из предыдущего с помощью некоторого переключения связей. Рассматриваемые в данной работе варианты переключения связей приведены ниже.

Значение $T(G)$ в процессе переключения связей может меняться. Обозначим через Δ_i изменение рассматриваемой величины в течении одного шага процесса – $\Delta_i = T(G_i) - T(G_{i-1}), 1 \leq i \leq k$. Для принятия или отклонения полученной сети применяется алгоритм Метрополиса-Гастингса, а именно – вероятность ω принятия переключения определяется следующим образом:

$$\omega = \begin{cases} 1, & if\ \Delta_i \geq 0 \\ e^{\mu * \Delta_i}, & if\ \Delta_i < 0 \end{cases} \qquad (18)$$

где μ – параметр модели: чем больше значение μ, тем меньше вероятность принятия сетей при отрицательном значением Δ_i. Очевидно, что все шаги, ведущие к росту величины $T(G)$, принимаются безусловно. Параметр модели μ имеет физический смысл - в термодинамике он соответствует понятию химического потенциала, используемого при описании состояния систем с переменным числом частиц и представляет собой энергию добавления одной частицы в систему без совершения работы.

Описанный процесс представляет из себя развитие сети в направлении роста $T(G)$; поэтому далее будем его называть *процессом направленного развития сети* или *эволюцией*.

Эволюция представляется *эволюционной траекторией*, т.е. изменением количества циклов длины три как зависимости от числа элементарных переключений (времени).

Сходимость длинных динамических траекторий к равновесному распределению было доказано Вигером и Латапи в 2005 году ([73]). В ходе изучения динамики нас интересуют равновесные состояния и, поэтому, динамический процесс продолжается до наступления сходимости. Под условием сходимости подразумевается продолжение эволюции до наступления *стабилизации* – состояния системы, при котором количество циклов длины три не подвергается значительным изменениям на достаточно длинном участке переключений, $\Phi(G, \mu)$, и система колеб-

лется (*флуктуирует*) около некоторого фиксированного значения. Естественно ассоциировать это состояние с равновесным состоянием.

Процесс переключения связей, основанный на некоторой стохастике, принято также называть *рандомизацией*. Данная динамика была предложена Масловым и Снипеном ([74]) и активно использовалась Алоном и др. ([75] [76]). Случай без топологических ограничений рассматривался в работах [77], потом [78], [79]. Случай с наложением топологических ограничений является новым направлением рассмотрения данной динамики – фазовый переход в системе при топологических ограничениях на мотивы был рассмотрен в 2015 году ([43]).

Сети, полученные в результате данного процесса, будем называть *финальными*.

Далее приводится описание двух рассматриваемых в данной работе вариантов переключения связей в течении процесса.

Процесс переключения без сохранения степеней

На каждом шаге процесса случайным образом (имеется в виду равномерное распределение по всем связям сети) выбирается одна связь - (u_1, v_1). Далее, из всех возможных пар несвязанных вершин случайным образом (равномерное распределение по всем указанным парам) выбирается одна пара – $\{u_2; v_2\}$. Связь (u_1, v_1) удаляется из сети, а пара $\{u_2; v_2\}$ – соединяется ребром. В результате данного переключения сохраняется общее количество связей в сети, но меняется топология. Далее приведен рисунок (рис. 5), показывающий одно такое переключение.

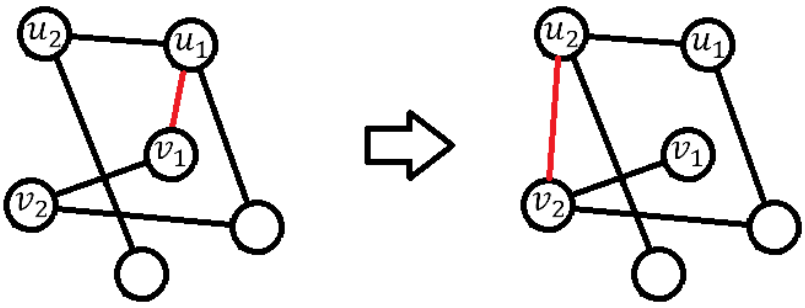


Рисунок 3. Переключение связей без сохранения степеней узлов

Процесс переключения с сохранением степеней

В данном варианте на каждом шаге случайным образом выбирается два ребра $\{(u_1, v_1); (u_2, v_2)\}$ при соблюдении следующего условия: в сети отсутствуют ребра (u_1, u_2) и (v_1, v_2). Далее два выбранных ребра удаляются из сети и добавляются ребра (u_1, u_2) и (v_1, v_2).

В результате данного переключения сохраняется не только общее количество ребер, но и распределение степеней. Ниже представлен пример подобного переключения (рис. 6).

Очевидно, что в данном примере условиям будет удовлетворять также добавление ребер

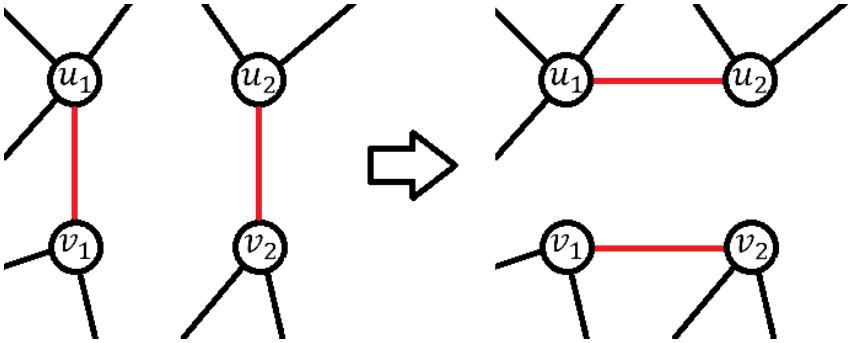


Рисунок 4. Переключение связей с сохранением степеней узлов

(u_1, v_2) и (v_1, u_2) после удаления изначально выбранных ребер.

2.4. Технологии параллельного и распределенного программирования

Двумя базовыми подходами для достижения одновременного выполнения составляющих частей программного обеспечения являются параллельное и распределенное программирование. Это две различные парадигмы программирования, у которых есть некоторые пересечения. Методы параллельного программирования предназначены для распределения работы выполняющейся программы между несколькими процессорами в контексте одной физической/виртуальной машины. Методы распределенного программирования предназначены для распределения работы программы между множеством процессов, причем выполнение различных процессов может производиться как на одной и той же машине, так и на разных. Иначе говоря, фрагменты распределенной программы в основном выполняются на различных компьютерах, имеющих каналы связи между собой, или по крайней мере в различных процессах. Программа, содержащая параллелизм, выполняется на одном и том же физическом или виртуальном компьютере. В этом случае программу можно разделить на процессы и/или потоки. Таким образом, многопоточность ограничивается параллелизмом. Формально говоря, параллельные программы иногда могут быть распределенными, например, в случае PVM-программировании (Parallel Virtual Machine — параллельная виртуальная машина). Распределенное программирование иногда может использоваться для реализации параллелизма, как, например, в случае с MPI-программированием (Message Passing Interface — интерфейс передачи сообщений). Однако не все распределенные программы могут включать параллелизм.

Фрагменты распределенной программы могут выполняться по различным запросам и в различные периоды времени. Например, программа календаря может быть разделена на две составляющие компоненты. Одна компонента будет обеспечивать пользователя информацией, присущей календарю, и форматом сохранения данных о его важных встречах, а другая компонента будет предоставлять пользователю набор оповещений для разных типов событий. Пользователь составляет расписание важных для него встреч, используя одну компоненту программного обеспечения, а другая его компонента выполняется независимо от первой. Набор оповещений и расписание пользователя вместе составляют единое приложение, которое разделено на две части, выполняемые по отдельности. При чистом параллелизме одновременно выполняемые компоненты являются частями одной и той же программы. Части распределенных приложений принято реализовать в виде отдельных программ ([80]). Типичная архитектура построения параллельных и распределенных приложений приведена на рис. 5.

Параллельное приложение, показанное в верхней части рис. 5, состоит из одной програм-

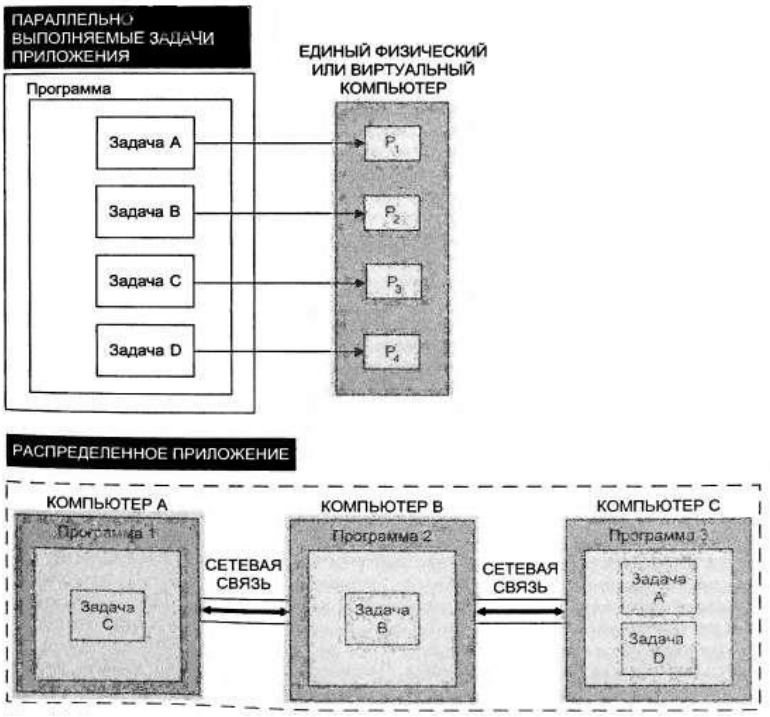


Рисунок 5. Типичная архитектура построения параллельной и распределенной программ

мы, разбитой на четыре задачи, которые выполняются на отдельных процессорах. Следовательно, все они могут выполняться одновременно. Эти задачи можно реализовать в виде распределенного приложения, показанного в нижней части рис 3. Оно состоит из трех различных программ, каждая из которых выполняется на отдельной машине. При этом программа 3 состоит из двух отдельных частей (задачи A и задачи D), которые выполняются на одной маши-

не. Несмотря на это, задачи A и D являются распределенными, потому что они реализованы как два отдельных процесса. Задачи параллельной программы более тесно связаны, чем задачи распределенного приложения. В общем случае процессоры, на которых выполняются распределенные программы, находятся на различных машинах, в то время как процессоры, на которых выполняются реализующие параллелизм программы, находятся на одном и той же машине. Конечно же, существуют гибридные приложения, которые реализуют и параллелизм, и распределение по различным машинам одновременно. Именно такие приложения постепенно становятся актуальными ([80]).

2.4.1. Параллелизация в контексте одного процесса. Технология OpenMP

Необходимость параллелизации в контексте одного процесса возникает в случае сильной связанности параллельных единиц по памяти. Применение процессов в данном случае приводит к большим накладным затратам на передачу информации между процессами и синхронизацию разделяемой памяти. Решением данной задачи является применение потоков. Потоки выполняются в контексте одного процесса и разделяют его память. Данный факт позволяет избежать накладных расходов на передачу данных и на их хранение в памяти. С другой стороны, механизм потоков позволяет управлять длительностью жизни потоков и динамически менять выполняемую задачу и, тем самым, повторно использовать потоки, задача которых уже выполнена. Применение потоков предполагает наличие параллелизма в решаемой задаче, иначе их использование не принесет результатов.

Потоки являются объектами операционной системы и управляются соответствующими системными вызовами. Создание, уничтожение и переключение между потоками является использованием ресурсов и представляют относительно дорогостоящие операции. Кроме того, существуют некоторые ограничения на количество одновременно существующих потоков в контексте одного процесса или в системе в общем. Поэтому, вместо многократного ручного создания и разрушения потоков, применяются техники, позволяющие сократить количество данных операций и свести их к минимуму. Основной техникой, выполняющей данную задачу, является использование пула потоков. Пул потоков является менеджером потоков и предоставляет соответствующий интерфейс для постановки новых задач в очередь выполнения. Основной идеей является повторное использование потоков, закончивших свою задачу. Пул изначально резервирует некоторое количество потоков, способных динамически обеспечить необходимую нагрузку, и, далее, запускает очередную доступную задачу на первом из свободных потоков. Количество потоков в пуле обычно задается соответственно количеству логических ядер в систе-

ме. Данное ограничение является существенным, так как число потоков, превосходящее количество доступных ядер, приведёт к частым переключениям между потоками и к спаду производительности не только приложения, но и системы в целом. Использование пулов потоков заключается в выделении параллельных участков алгоритма в отдельные потоки и их выполнение в пуле. В некоторых операционных системах пулы тоже являются системными объектами и предоставляют соответствующие интерфейсы. Создание пулов в системах, не поддерживающих их на системном уровне, предполагает реализацию следующих функций: выделение потоков, постановка задач на исполнение с последующей обработкой и уничтожение потоков. Основная часть языков программирования имеет поддержку потоков и пулов потоков либо в составе стандартной библиотеки, либо в виде готовых библиотек, распространяющихся отдельно.

Язык программирования Си++ имеет встроенную поддержку потоков, но не содержит реализации пула потоков в стандартной библиотеке. Однако, все необходимые средства для создания пулов существуют, и сама реализация не представляет сложности. Кроме того, доступно множество библиотек, реализующих необходимую функциональность.

Другим подходом для параллелизации является использование технологии OpenMP. Стандарт является открытым и определяет принципы написания параллельных программ, предназначенных для вычислительных систем с множеством процессоров, имеющих общую оперативную память. Приложение организовывается как множество потоков, имеющих общую память, а проблема синхронизации доступа к памяти решается использованием стандартных методов, таких как критические секции и мониторы.

Стандарт OpenMP был представлен в 1997 году, как программируемый интерфейс приложения (API), ориентированный на написание переносимых многопоточных приложений. Изначально стандарт был ориентирован на язык Fortran, но позднее был расширен и включил языки Си и Си++ ([81]).

Стандарт разрабатывается и поддерживается организацией OpenMP Architecture Board (ARB), в которую входят представители многих крупнейших разработчиков программного обеспечения и SMP-архитектур. В 1997 и 1998 годах были представлены спецификации для языков Фортран и Си/Си++ соответственно. OpenMP спроектирован как стандарт разработки высокопроизводительных приложений для масштабируемых SMP-систем, основанных на модели общей памяти. На момент написания данной работы последней официальной спецификацией стандарта является версия 4.5.

OpenMP предоставляет три вида средств управления потоком исполнения программы: (i) набор специальных директив препроцессора, (ii) множество библиотечных функций и (iii) переменные окружения. Наиболее распространенными являются директивы препроцессору, которые используются для обозначения фрагментов кода с поддержкой параллельного выполнения. Компилятор, имеющий поддержку OpenMP, обрабатывает исходный код программы и заменяет директивы параллельного выполнения областей кода соответствующими вызовами функций ([81]).

Идея “частичного распараллеливания” делает OpenMP идеально подходящим для разработчиков, которые желают распараллелить свои вычислительно сложные программы, содержащие большие параллельные циклы. Программист не разрабатывает новую программу, реализующую параллелизм, а просто добавляет в текст уже написанной последовательной программы соответствующие OpenMP директивы. Предполагается, что программа с поддержкой стандарта в случае выполнения на платформе с одним процессором может выполняться в качестве последовательной, т.е. отсутствует необходимость поддержки двух версий одной и той же программы - последовательной и параллельной. Директивы OpenMP, как и любые другие неподдерживаемые директивы, игнорируются последовательным компилятором (т. е. компилятором, не поддерживающим стандарт OpenMP), а для вызова процедур OpenMP могут быть использованы специальные заглушки, определения которых приводятся в спецификации стандарта.

Стандарт OpenMP достаточно прост в использовании и включает два базовых вида конструкций: pragma-директивы и набор функций исполняющей среды OpenMP. Директивы pragma, как правило, указывают препроцессору компилятора как реализовать параллельное выполнение участков кода. Директивы начинаются с фразы pragma omp. Компилятор, не поддерживающий конкретную технологию, такую как OpenMP, просто игнорирует соответствующие директивы pragma. Каждая из директив может иметь дополнительные атрибуты. Отдельно определяются атрибуты для назначения классов переменных, которые могут быть атрибутами различных директив. Функции стандарта предназначены в основном для изменения и получения переменных окружения. Кроме того, стандарт предоставляет API-функции для поддержки некоторых типов синхронизации.

Основные достоинства

- Эффективное использование возможностей систем с общей памятью.
- Возможность поэтапного (инкрементального) распараллеливания последовательных программ.

- Единственность разрабатываемого кода: нет необходимости поддерживать два варианта программы - последовательный и параллельный, поскольку директивы игнорируются обычными не поддерживающими стандарт компиляторами.
- Переносимость программ, поддержка стандарта в наиболее распространенных языках (Си/Си++, Фортран) и ОС (Windows, Unix-подобные).

2.4.2. Распределенные вычисления на основе технологии CUDA

Существует тенденция использования графических карт для решения широкого спектра задач из различных областей – квантовой механики, нейронных сетей, искусственного интеллекта, криптографии, различных физических и математических расчетов, реалистичной визуализации объектов, их реконструкции и распознавания по фотографиям и т.п. Эти задачи достаточно трудно решать в рамках существующих программных графических интерфейсов (DirectX, OpenGL), так как последние создавались для решения совсем других задач.

Возникновение новой технологии CUDA (Compute Unified Device Architecture), нацеленной на более широкий чем растеризация круг задач, обусловлено развитием программирования общего назначение с использованием графических адаптеров (General-purpose computing on graphics processing units, GPGPU). Технология была разработана компанией NVidia. Основной конкурирующей является технология STREAM компании ATI. Однако технология STREAM сильно отстаёт в развитии от CUDA, и, поэтому, пользователи в основном используют CUDA – технологию GPGPU, позволяющую разрабатывать приложения на подмножестве языков Си/Си++ ([82]).

Далее вкратце представлены некоторые существенные различия областей и особенностей применения центрального процессора (CPU – Central Processing Unit) и видеокарты.

Процессор изначально предназначен для решения задач общего типа и имеет доступ к произвольной адресуемой памяти. Программы, работающие на центральном процессоре, могут обращаться напрямую к произвольным ячейкам линейной и однородной памяти (в рамках механизма организации памяти конкретной операционной системы).

Для GPU это не так. В CUDA имеется пять видов памяти – локальная память, глобальная память, разделяемая память, константная память и текстурная память. Кроме того, как и в случае центрального процессора, имеются регистры. Читать разрешается из любой ячейки, доступной физически, однако записывать разрешается только в определенные ячейки. Причина подобного ограничения заключается в том, что GPU является специфическим устройством,

предназначенным для конкретных целей. Это ограничение позволяет увеличить скорость работы некоторых алгоритмов и, кроме того, способствует снижению стоимости оборудования.

В основе вычислительной архитектуры CUDA лежит понятие мультипроцессора и концепции одна команда на множество данных (SIMD - Single Instruction Multiple Data). Эта концепция предполагает одновременную обработку множества данных при выполнении одной инструкции. Мультипроцессор представляет из себя многоядерный SIMD процессор, позволяющий в каждый фиксированный момент времени выполнять на всех ядрах только одну инструкцию. Каждое ядро такого процессора является скалярным и не поддерживает векторные операции в чистом виде.

Под термином устройство (device) принято понимать видеоадаптер, поддерживающий соответствующий драйвер CUDA, или некоторое другое специализированное устройство, разработанное с поддержкой соответствующих технологий и предназначенное для исполнения приложений, использующих CUDA (примером такого устройства является NVIDIA Tesla).

Термином хост (host) обозначается программа в основной оперативной памяти компьютера, выполняемая на центральном процессоре и содержащая управляющие функции работы с устройством. Проще говоря, тот фрагмент программы, который работает на центральном процессоре именуется хостом, а видеокарта - устройством. Абстрактно устройство можно представить как множество мультипроцессоров и драйвер CUDA.

Процедура, выполняемая на устройстве, называется *ядром.*

Важной особенностью архитектуры CUDA является блочно-сеточная организация, не являющаяся характерной для многопоточных приложений. Драйвер CUDA самостоятельно распределяет доступные ресурсы устройства между выполняемыми потоками.

Все потоки, выполняющие ядро, объединяются в блоки, а блоки, в свою очередь, объединяются в сетку. Идентификация потоков производится посредством двухмерных индексов. Разработчики CUDA предусмотрели возможность работать с одномерными (простыми), двухмерными или трехмерными индексами, в зависимости от ситуации и удобства программиста. В общем случае индексы являются трехмерными векторами. Для каждого потока доступен индекс потока внутри блока – *threadIdx,* и индекс блока внутри сетки – *blockIdx.* При запуске все потоки отличаются только значениями этих индексов. Фактически, именно посредством этих индексов программист управляет выполнением программы, определяя, какая именно часть данных обрабатывается в каждом из потоков ([82]).

Основные преимущества и ограничения

Архитектура CUDA обладает следующими преимуществами:

- Основным интерфейсом программирования приложений CUDA (CUDA API) является язык программирования Си с некоторыми ограничениями, что позволяет упростить и сгладить процесс знакомства и изучения архитектуры CUDA.
- Эффективные транзакции между памятью центрального процессора и видеопамятью.
- Полная аппаратная поддержка целочисленных и побитовых операций.

Однако стоит привести следующие ограничения:

- Ядра не поддерживают рекурсии и имеют некоторые другие ограничения.
- Архитектуру CUDA поддерживается и развивается только компанией NVidia.

2.4.3. Кластерные вычисления на основе технологии MPI

Для вычислительных систем с распределенной памятью свойственна независимая друг от друга работа процессоров. Организация параллельных вычислений в этом случае предполагает распределение вычислительной нагрузки по различным процессам и организацию передачи информации между ними. Для решение приведенных вопросов используется интерфейс передачи сообщений (MPI – message passing interface) ([83]).

Для организации распределения вычислительных операций между процессорами необходимо выполнить следующие шаги:

1. провести анализ алгоритма решения задачи,
2. выявить независимые фрагменты вычислений,
3. реализовать программное представление этих фрагментов,
4. распределить полученные части программы по разным процессорам (машинам).

Стандарт MPI для решения описанной задачи применяет более простой подход, а именно – разрабатывается единственная программа, которая запускается на выполнение одновременно на всех имеющихся процессорах. При этом для обеспечения различия в вычислениях на разных процессорах, можно

1. использовать различные данные для программ, работающих на разных процессорах
2. использовать имеющиеся в стандарте MPI методы идентификации процессора, выполняющего программу (таким образом, поддерживается возможность получения различных вычислений в зависимости от выполняющего программу процессора).

Подобный механизм организации параллельных вычислений называется моделью "одна программа – множество процессов" (SPMP – single program multiple processes).

Для организации взаимодействия между процессами в минимальном варианте достаточна поддержка базовых операций пересылки данных (предполагается, что должны существовать технические средства организации связи между процессорами – линии связи). Стандарт MPI предусматривает поддержку большого множества операций передачи информации между процессами. Таким образом реализуются различные методы пересылки данных. Именно широкие возможности взаимодействия между процессами являются сильной стороной стандарта MPI (о чём, в частности, свидетельствует название MPI) ([83]).

К появлению стандарта MPI способствовало рабочее совещание по стандартам для передачи сообщений в среде с распределенной памятью в апреле 1992 года. Результатом этого совещания стало создание рабочей группы, которая позднее была преобразована в сообщество «MPI Forum». Последующая деятельность сообщества привела к созданию и принятию в 1994 году версии 1.0 стандарта MPI. В дальнейшем стандарт MPI активно развивался, и в 1997 году была принята версия 2.0, а в 2012 году –версия 3.0. Стандарт является открытым и доступен на официальном веб-сайте сообщества (http://mpi-forum.org).

Основные преимущества и недостатки

Далее представлены некоторые важные аспекты стандарта MPI ([83]):

- MPI позволяет значительно облегчить разработку переносимых параллельных программ между разными платформами – параллельная программа, реализованная на языке Си или Фортран с использованием библиотеки MPI, будет работать на разных вычислительных платформах благодаря наличию соответствующих библиотек реализации;
- MPI способствует повышению эффективности вычислений, поскольку практически для всех типов вычислительных систем существуют реализации библиотек MPI наиболее оптимально использующих возможности доступного оборудования;
- MPI упрощает разработку параллельных программ, так как, большая часть основных операций пересылки сообщений предусматривается стандартом MPI, и, кроме того, доступно множество библиотек параллельных методов, созданных с использованием MPI;
- MPI повышает масштабируемость параллельных программ.

Кроме того, следует упомянуть некоторые недостатки стандарта MPI:

- MPI является низкоуровневым инструментом программиста.
- Большинство реализаций MPI в полной мере не поддерживают совмещение передачи сообщений с вычислениями.

- MPI не предусматривает возможности задания стартового размещения процессов по процессорам.
- Отладка программ затрудняется одновременным исполнением нескольких программных ветвей.

Далее приведены понятия и определения, являющиеся основными в стандарте MPI.

Понятие параллельной программы

Параллельная программа в MPI представляет из себя множество одновременно выполняющихся процессов, созданных на основе единого программного кода. Различные процессы могут выполняться на разных процессорах. Кроме того, на одном процессоре могут располагаться несколько процессов (исполнение в этом случае производится в режиме разделения времени и обеспечивается локальной операционной системой). В предельном случае параллельная программа может выполняться на одном процессоре – такой способ часто применяется для начальной проверки корректности параллельной программы.

Каждый из процессов параллельной программы запускается посредством копирования единого программного кода. Этот код, представленный в виде исполняемого файла, в момент запуска приложения должен быть доступен на всех используемых процессорах. Для разработки исходного программного кода используются алгоритмические языки программирования Си/Си++ или Фортран с применением некоторой библиотеки реализации MPI ([83]).

Общее количество процессов и число используемых процессоров/машин задается в момент запуска параллельной программы средствами среды исполнения MPI-программ. В процессе вычислений количество может меняться при помощи применения специальных, но редко используемых средств динамического создания процессов и управления ими, появившихся в версии 2.0 стандарте MPI. Все процессы работающей программы последовательно пронумерованы от 0 до $n-1$, где n – число процессов, заданных в момент запуска. Номер процесса, уникально идентифицирующий его, принято называть рангом процесса.

Операции передачи данных

В основе стандарта MPI лежат операции передачи сообщений. Поддерживаемые в MPI функции можно разделить на *парные* (*point-to-point*) операции между двумя процессами и *коллективные* (*collective*) операции для одновременного взаимодействия множества/группы процессов.

Наиболее распространенными среди режимов передачи данных во время выполнения парных операций являются синхронный и блокирующий режимы. Кроме того, в стандарт MPI включено большинство основных коллективных операций передачи сообщений.

Понятие коммуникатора

MPI предоставляет возможность объединения процессов параллельной программы в группы. Другой важной возможностью MPI, позволяющей описывать набор процессов, является понятие *коммуникатора*. Коммуникатор в MPI - специальный служебный объект, предназначенный для объединения множества процессов и, кроме того, ряд дополнительных параметров (называемых контекстом), используемых во время выполнения операций передачи данных ([83]).

Выполнение парных операций передачи данных ограничено только теми процессами, которые принадлежат одному коммуникатору. Коллективные операции осуществляются одновременно для всех процессов одного коммуникатора. Как следствие, при передаче данных в MPI необходимо указывать используемый коммуникатор.

Новые и существующие группы процессов и коммуникаторы могут создаваться и удаляться динамически во время выполнения вычислений. Процесс может одновременно принадлежать разным группам и коммуникаторам. Все процессы в параллельной программе входят в состав создаваемого по умолчанию коммуникатора, имеющего идентификатор MPI_COMM_WORLD.

Начиная с версии 2.0 в стандарт MPI была добавлена возможность создания глобальных коммуникаторов (*inter-communicator*), и объединять в одну структуру пару групп для выполнения коллективных операций между процессами из разных групп.

Типы данных

Функции MPI, предназначенные для выполнения операций передачи сообщений требуют указания типа пересылаемых данных. Стандарт MPI содержит большое количество базовых типов, которые во многом совпадают с типами данных в языках программирования Си и Фортран. Кроме того, MPI поддерживает создание новых производных типов данных для более точного и краткого описания пересылаемых данных.

Виртуальные топологии

Как уже было отмечено, выполнение парных операций передачи данных ограничено процессами одного коммуникатора, а в случае коллективной операции - всеми процессами, входящими в коммуникатор. Логическая топология линий связи между процессами является полным

графом (независимо от реальной физической топологии, т. е. наличия реальных физических каналов связи между процессорами).

Наряду с этим, для изложения и последующего анализа некоторых параллельных алгоритмов представляется целесообразным логическое представление имеющейся физической коммуникационной сети в виде различных, не совпадающих с физической, топологий.

В MPI поддерживается представление множества выполняемых процессов в виде решеток с произвольной размерностью. При этом процессы, которые расположены на краях решеток, могут быть объявлены соседними, и, как следствие, на основе решеток могут быть определена структура типа тор ([83]).

Кроме того, стандарт MPI поддерживает средства для конструирования логических топологий, имеющих любой заданный тип.

ГЛАВА 3: РАЗРАБОТАННАЯ ПРОГРАММНАЯ СРЕДА

Глава представляет программную среду, разработанную для проведения исследований. Раздел 3.1 представляет основные требования к системе, и задачи, которые система должна позволять решать. Раздел 3.2 представляет объём вычислений при исследовании процесса направленного развития сетей. Раздел 3.3 приводит анализ параллельных и распределённых технологий, проделанный при создании системы. Раздел 3.4 представляет разработанные алгоритмы эволюции для двух рассматриваемых вариантов переключения связей. Раздел 3.5 представляет основные этапы процесса имплементации системы. Раздел 3.6 описывает архитектуру системы на различных уровнях – начиная от высокоуровневой архитектуры и заканчивая структурой ядра и потока исполнения. Раздел 3.7 приводит временные характеристики системы и некоторые её особенности.

3.1. Основные требования

Анализ поставленных задач исследования позволил выявить основные требования к программной среде:

1) Возможность моделирования процесса направленного развития с поддержкой двух описанных ранее вариантов. Данное требование является основным, так как ведется исследование процесса развития сетей при определенных топологических ограничениях.
2) Возможность работы с сетями больших размеров и с ансамблями сетей

   Получаемые результаты необходимо подтверждать либо на сетях больших (в теории, стремящихся к бесконечности) размеров либо на ансамблях более маленьких сетей (данное свойство динамических систем называется *эргодичностью* [84]). Поэтому необходимо иметь возможность работы как минимум на одном из указанных вариантов и желательно – на обоих.
3) Возможность получения длинных (достигающих стабилизации) траекторий

   Траектории являются начальным материалом для анализа свойств процесса направленного развития. Для получения общей картины анализ необходимо проводить в стабильных состояниях, для достижения которых может потребоваться много времени. Поэтому, для корректного анализа наблюдаемых явлений необходима поддержка траекторий, достигающих стабилизации.
4) Возможность получение сетей по ходу процесса

Траектории дают только начальное представление о процессе. Для дальнейшего анализа необходимо иметь состояния сетей по ходу процесса в определенных точках. Важным требованием для данной опции является отсутствие значительного воздействия на общее время выполнения.

5) Возможность графического представления полученных данных

   Сырые данные содержат всю необходимую информацию, но она недоступна без специальных манипуляций с данными. Основной из таких манипуляций является удобная визуализация полученных данных – таких, как, например, изображение сети на плоскости с помощью силовых алгоритмов, позволяющих раскрыть структуру сети, или же изображение спектральной плотности матриц смежности в виде гистограммы. Желательно получать графические данные без каких-либо сторонних программных продуктов, тем более, что некоторые из них являются коммерческими и недоступны для свободного пользования.

6) Оптимальное использование ресурсов

   При получении результатов важным является то, чтобы с одной стороны пользователю не пришлось ждать слишком долго для получения некоторого результата, а с другой стороны – в случае разделяемой несколькими пользователями среды – использование ресурсов позволяло одновременно работать нескольким задачам различных пользователей. Поэтому оптимальное использование системных ресурсов – в первую очередь процессора и оперативной памяти – требует применения соответствующих техник и является важным требованием.

7) Удобный интерфейс пользователя

   Данное требование является опциональным, так как работа на данной системе предполагает некоторые знания соответствующих технологий и методов работы с ними. Желательно, чтобы система предоставляла легкие в использовании команды си опции. Кроме того, желательно чтобы команды предоставляли возможность получения описаний производимых действий и содержали информацию о доступных опциях и выполняемых ими действий.

Основной задачей разработанной системы является всестороннее исследование процесса направленного развития сетей. Система должна предоставлять средства для получения следующих данных:

1) Эволюционные траектории
2) Основные свойства сетей
3) Структурная картина сетей
4) Свойства матриц смежности сетей

Кроме того, система, как уже было сказано ранее, должна предоставлять возможности выполнения манипуляций с данными и, в частности, визуализации данной информации.

3.2. Объём вычислений и необходимость специализированной системы

Первый этап исследования заключается в определении и изучении критических точек параметра μ (см. раздел 2.3). На данном этапе выполняются следующие шаги:

5) Нахождение критического порога параметра μ.
6) Построение траекторий вблизи критического порога.
7) Анализ стабильных хвостов траектории.
8) Анализ свойств сетей
9) Получение результатов для различных значений параметров.

Первый шаг предполагает для каждого фиксированного начального состояния (N, p) просмотр широкого интервала значений параметра μ с целью выявления критического порога. В интервале равномерно выбираются несколько значений параметра и для каждого из них генерируется траектория. Количество шагов, в зависимости от начальных параметров, может меняться от нескольких миллионов до нескольких десятков миллионов шагов. Данный процесс является итеративным. На каждой следующей итерации, в соответствии с результатами предыдущей итерации, рассматривается либо сдвинутый, либо более узкий интервал значений для получения значения критического порога с необходимой точностью.

Генерация траекторий без применения параллелизации – процесс долгий. Для сети с параметрами $N = 256, p = 0.3$ и траектории длиной 10 миллионов шагов такая генерация длится порядка часа (все приведенные данные по времени выполнения получены на процессоре Intel Core i7 шестого поколения). Рассмотрим примененные техники на примере приближенного времени расчета критического значения параметра в случае сети с упомянутыми параметрами. Предположим проводится 3 итерации упомянутого алгоритма и на каждом шаге рассматривается 10 траекторий. Такой процесс позволит получить значение параметра с точность порядка нескольких сотых долей единицы. Кроме того, получение критической точки для фиксированных значений входных параметров необходимо подтвердить на ансамбле сетей. Предположим размер ансамбля равен 20.

Итак, необходимо последовательно сгенерировать 3 раза по 10 траекторий для 20 сетей (600 траекторий). Без какой-либо параллельности (т.е. без применения разработанных методов на основе параллельных вычислений) общее время расчетов составляет 600 часов (или, 25

дней). Ясно, что эта величина неприемлема для проведения исследований и необходима разработка и применение отвечающих временным требованиям вычислительных методов и программных решений. Основной целью разработки новой системы является быстрая генерация длинных траекторий для ансамбля сетей и получение всех необходимых свойств без каких-либо дополнительных манипуляций с данными и переключения между различными программными средами.

Первый подход, примененный в разработанной системе для решения данной задачи – одновременное выполнение генерации траекторий для различных значений параметра модели и для различных сетей. В приведенном примере – параллельная генерация 10 траекторий для 20 сетей (200 траекторий) на каждой итерации – позволит, в идеальном случае назначения каждому процессу отдельного ядра, сократить время генерации в 200 раз. Однако локальные компьютеры ресурсами таких объёмов не обладают. В основном, кластеры также не обладают подобным количеством доступных ядер. Ввиду этого, в системе запроектирована ключевая возможность работы на заданном количестве ядер. Траектории равномерно распределяются по очередям, каждая из которых выполняется на выделенном для этой очереди ядре. Результат – при наличии k ядер, общее время выполнения сокращается в $\left\lceil \frac{200}{k} \right\rceil$ раз (в описанном случае). Учитывая независимость отдельных процессов по входным данным, можно запускать параллельные процессы на различных машинах. Известной и распространенной технологией, позволяющей выполнять подобное распределение задач по вычислительным сетям (кластерам) является стандарт MPI. Стандарт определяет основные аспекты управления процессов и методы организации взаимодействия между ними. Существуют различные библиотеки реализации стандарта и различные кластеры используют различные реализации. Для абстракции от конкретной реализации в разработанной системе и возможности легкого переноса кода с одной конфигурации на другую, система спроектирована с использованием оболочки, использующей только методы стандарта и имеющей возможность использовать различные библиотеки реализации без значительной потери производительности.

Применение упомянутой техники позволяет снизить время выполнения до 3 часов в случае назначения каждому набору параметров одного ядра. Однако, данная величина также является достаточно большой с точки зрения обычного пользователя.

На следующем этапе были рассмотрены параллельные технологии для увеличения скорости генерации одной траектории. Анализ возможности применения технологии CUDA выявил её

непригодность в контексте процесса направленного развития ввиду последовательной природы последнего. Поэтому были детально проанализированы действия на каждом шаге направленного развития случайных сетей и были выявлены критические для производительности участки. Далее были разработаны соответствующие алгоритмы, позволяющие уменьшить общее время выполнения за счет уменьшения количества действий и параллельного выполнения некоторых из них. Результатом данных модификаций стало сокращение времени расчета одной траектории в рассматриваемом случае до 10-12 минут, что является ускорением в 5-6 раз.

Итак, финальное время вычисления критического порога в рассмотренном случае при наличии соответствующей аппаратной составляющей составляет 30-40 минут, и это является приемлемым временем расчета (учитывая использование ансамбля из 20 сетей). Зависимость времени выполнения от входных параметров представлена в разделе 2.7.

Второй этап исследования предполагает получение траекторий около критического порога и анализ свойств сетей. Для этого необходимо иметь состояние сети в необходимые моменты времени и, следовательно, соответствующая функциональность должна поддерживаться программной средой. Количество сетей, необходимых для проведения исследований, зависит от длины траектории и величины интервала времени, через которые необходимо сохранять состояние сети. Важным аспектом сохранения сетей является быстрота производимых действий – сохранение не должно значительным образом влиять на время генерации траекторий. Данное требование учтено при конструировании программной системы.

Объём вычислений на следующих этапах исследований меньше. Основные задачи – получение и графическое представление траекторий, свойств, сетей, матриц смежности, спектров и спектральных плотностей матриц смежности. Данные возможности реализованы в системе при помощи применения известных алгоритмов, адаптированных и модифицированных для работы с остальными частями системы.

3.3. Параллельные и распределенные технологии в контексте рассматриваемой задачи

Для разработки и имплементации системы были проанализированы распространенные программные технологии параллельного и распределенного выполнения. Прежде чем привести данный анализ, опишем основные входные/выходные данные и параметры программной модели, а также некоторые специфики работы алгоритмов.

Входом процесса является некоторая фиксированная сеть, набор значений параметра модели, для которых необходимо сгенерировать траектории, а также шаг, через который надо сохранять состояние сети. На выходе должны быть получены траектории для каждого значения входного параметра и состояния сетей в заданные моменты времени.

Процесс эволюции сам по себе имеет последовательную природу и поэтому сложно поддается параллелизации (примененные методы, позволившие повысить производительность, представлены ниже в разделе «Алгоритмы эволюции»). Ввиду сказанного, основными рассмотренными аспектами при параллелизации являются:

- Параллельное выполнение процессов для каждого значения входного параметра

  Данный аспект предполагает наличие параллельных или распределенных единиц для выполнения процесса на одном фиксированном значении входного параметра. Очевидным подходом является применение процессов для выполнения данной задачи. Однако в таком случае возникают вопросы взаимодействия между ними, что не является частью требований к системе. Поэтому необходимо применение соответствующих технологий, предоставляющих необходимую функциональность. Подробный анализ данного аспекта приведен ниже – подзаголовок «Анализ технологии MPI»
- Параллельная обработка компонентов сетей на каждом шаге

  Данный аспект предполагает использование мультипроцессоров для одновременного выполнения множества однотипных действий на множестве входных данных. Однако, это может привести к большим накладным расходам. Подробный анализ данного случая представлен ниже - подзаголовок «Анализ технологии CUDA»

3.3.1. Анализ технологии MPI

Технология MPI основывается на идее параллельных процессов, общающихся по заранее определенному протоколу. Параллельные процессы, как было представлено в главе 1, могут выполнятся как на одной машине, так и на удаленных компьютерах, объединённых в кластеры. Это позволяет абстрагироваться от реальной машины и, при разработке приложения, представлять вычислительную среду в виде компьютера с необходимым количеством процессоров. Взаимодействие между процессами, которое в случае работы на кластере производятся при помощи сетей, для программиста является прозрачным, так как он работает с высокоуровневой абстракцией реальной компьютерной системы и не задумывается о наличии каналов связи или же о реальной топологии компьютерной сети, объединяющей компоненты кластера.

Основными причинами использования технологии MPI являются:

1) Масштабируемость от параллельных компьютеров до кластеров

   Как уже было сказано, данное свойство позволяет не задумываться о реальной системе, на которой будет выполнятся приложение, а иметь дело только с его некоторым абстрактным аналогом.

2) Переносимость между множеством платформ

   Данное свойство позволяет также не задумываться о реальной среде, в которой будет исполняться приложение, а опять-таки иметь дело с абстракцией.

3) Различные методы взаимодействия между процессами

   Стандарт MPI предоставляет множество методов синхронной и асинхронной передачи данных (точка-точка, широковещание, коллективные методы), что дает гибкости разработчику. Также стандарт предоставляет методы обширные методы синхронизации процессов, что позволяет реализовать сложные коммуникационные алгоритмы.

4) Автоматическое и эффективное управление процессами

   С большой степенью абстракции MPI можно рассматривать как менеджер процессов, целью которого является обеспечение всех вышеупомянутых требований. Реальные механизмы управления возложены на различные имплементации, однако требование к эффективности является ключевым.

3.3.2. Анализ технологии CUDA

Технология CUDA, как уже было представлено в главе 1, является SIMD-технологией. Поэтому, максимальное быстродействие обеспечивается при наличие достаточно большого числа потоков (которые в случае технологии CUDA являются достаточно облегченными вариантами потоков обычного процессора и поэтому их создание не является такой же тяжелой функцией, какой является создание потока на процессоре), которые работают над некоторым участком больших данных. Действия, выполняемые каждым потоком, являются идентичными – написания какого-либо дополнительного кода не требуется. Следовательно, для создания большого количества потоков необходимо наличие большого количества данных, на которых будет выполнен один и тот же алгоритм.

При рассмотрении процесса направленного развития сетей, как было представлено в требованиях к системе, важным является рассмотрение множества значений параметра модели. Ясно, что при различных значениях параметра необходимо запускать отдельные процессы –

связи между задачами двух различных процессов нет. Поэтому, необходимо анализировать применимость технологии CUDA в рамках одного процесса. Как уже было сказано выше, процесс направленного развития является последовательным и, как следствие, основным для производительности участком является шаг рандомизации – выбор связей/вершин, переключение и подсчёт изменения количества циклов длины три.

Рассмотрим шаг процесса и возможности применения технологии CUDA. Основным источником данных для одновременного рассмотрения на каждом шаге являются компоненты сети. Далее представлены основные варианты:

1) Использование вершин сети

   При данном подходе каждому потоку назначается некоторая вершина сети и всю работу по обработке вершины выполняется этим потоком. Для выбора ребер и вершин на каждом шаге процесса эволюции выделятся один поток, который также будет принимать решение по принятию/отклонению очередного шага процесса. На каждом шаге процесса данный поток делает соответствующий выбор и активирует потоки, обрабатывающие выбранные вершины и их непосредственных соседей (рассматривать вершины, удаленные на большее расстояние в случае рассмотрения циклов длины три смысла не имеет, потому что остальные вершины не могут составлять упомянутого типа циклов с выбранными вершинами). Ясно, что каждый из потоков в периоды времени, пока выделенный поток выбирает ребра и/или вершины для очередного переключения, а также во время принятия/отклонения шага, находятся в режиме ожидания. Это значительно влияет на общую производительность процесса, так как основную часть времени многие потоки ожидают.

2) Использование ребер сети

   Другим подходом является рассмотрение ребер сети. В данном случае ввиду того, что количество ребер в основном значительно больше чем вершин, каждому потоку необходимо назначить некоторое множество ребер сети. Опять же, необходим выделенный поток, делающий начальный выбор ребер/вершин на каждом шаге и принимающий решение о принятии/отклонении шага. Однако, в отличии от первого случая, где множество вершин не меняется, множество ребер в ходе процесса меняется, из-за чего необходимо назначение новым ребрам потоков. Это может быть сделано простым обновлением тех потоков, которые обрабатывали удаленные ребра. В данном случае на каждом шаге потоки рассматривают множество ребер. Для получения конечного результата, не-

обходимо совмещать результаты потоков, что требует значительных накладных расходов процессорного времени. В результате получается, что в ходе работы больше времени тратится на накладные расчеты и операции, нежели на полезные действия - это делает данный вариант неприменимым.

3) Рассмотрение всех мотивов с подходящей топологией

   В данном варианте рассматривается множество мотивов, топология которых удовлетворяет поставленным требованиям. Количество таких мотивов большое и их количество на каждом шаге меняется. Расчет изменений мотивов в данном случае уже не является локальной задачей, поэтому для их нахождения требуется обход всех ребер сети. Так как количество вычислений в данном варианте значительно больше, чем в предыдущих, ясно, что данный алгоритм также неприменим.

3.4. Алгоритмы эволюции

Далее будут описаны алгоритмы рандомизации, разработанные и реализованные в системе ([3]). При разработке данных алгоритмов были выделены действия, не зависящие по данным, которые возможно выполнять параллельно; были проанализированы различные варианты выполнения каждого шага рассматриваемого процесса эволюции сети и применены наиболее оптимальные по времени решения.

Сначала, представлен общий вид процедуры программного моделирования эволюции сети (алгоритм 2). На вход данный алгоритм получает сеть и значения параметра μ. На выходе алгоритм выдает эволюционную траекторию – последовательность наблюдаемого свойства (количества циклов длины три) по временной шкале. Алгоритм на первом шаге вычисляет начальное значение рассматриваемого свойства (например, циклы длины три) и далее обновляет значение по ходу следующих шагов. Процесс рандомизации продолжается до наступления стабилизации. Были рассмотрены два подхода для получения траектории со стабильным хвостом. Первый – наблюдение за достаточно большим участком траектории с целью вычисления величины разброса значений. При достаточно малом значении разброса на необходимом количестве шагов, процесс останавливается. Данный подход требует значительных вычислений на каждом шагу процесса, вследствие чего увеличивается общее время моделирования. Поэтому применяется второй, более быстрый подход – ручное определение количества необходимых шагов выполнения процесса и прогонка алгоритма рандомизации с полученным значением. Данный подход требует дополнительные прогоны алгоритма для определения начальных параметров,

однако, в отличие от первого, не требует накладных расчетов при генерации каждой траектории.

3.4.1. Алгоритм переключения с сохранением степеней вершин

Первым представлен разработанный алгоритм переключения связей с сохранением степеней вершин. Алгоритм 3 представляет последовательность действий, производимых на каждом шаге процесса в данном случае. В начале выбираются два случайных ребра при условии, что концы ребер не связаны (смотри подробное описание процесса в главе 2). Случайные ребра генерируются равномерно из множества всех ребер графа. Выбранные ребра удаляются и, вместо них, добавляются два новых ребра, соединяющих вершины удаленных ребер в ином порядке. Далее рассчитывается изменение наблюдаемого свойства и, на основе формулы вычисления вероятности принятия шага, проделанный шаг либо принимается, и измененная сеть поступает на вход следующего шага процесса, либо же измененная сеть отбрасывается и, слеДующий шаг процесса получает входную сеть текущего шага. В конце, данная процедура возвращает изменение значения наблюдаемого свойства.

```
1: PROCEDURE Randomization(G; m)
2:     trajectory = []
3:     prop = CalculateObservedProperty(G)
4:     trajectory.append(prop)
5:     WHILE !Stabilized(trajectory) DO
6:         D_prop = RandomizationStep(G; m)
7:         prop += D_prop
8:         trajectory.append(prop)
9:     RETURN trajectory
```

Алгоритм 2. Общий алгоритм рандомизации

Данный алгоритм содержит два основных фрагмента, которые имеет смысл распараллеливать. Первый – генерация случайного ребра (процедура *GetRandomEdge()*). Речь идёт не о самом процессе генерации, а о параллельной генерации нескольких ребер. Применение пула потоков решает данный вопрос. Второй участок – подсчет изменения величины наблюдаемого свойства (процедура *CalculatePropertyChange()*). В данном случае необходим подсчёт добавленных и удалённых циклов в сети как результат произведенных переключений. Данная опе-

рация является локальной и поэтому необходимо рассмотрение только непосредственных соседей рассматриваемых вершин. Кроме того, параллельное рассмотрение каждой вершины в отдельном потоке значительно ускорит процесс.

Далее представлен алгоритм подсчета изменения количества циклов длины три в сети при переключении связей с сохранением степеней вершин (алгоритм 4). Параллельно выполняются четыре задачи – по одной на каждое удаляемое и каждое добавляемое ребра, причем удаляемые ребра удаляются заранее, а добавляемые – не добавляются, а только рассматривается как добавленное. Последний факт позволяет ещё больше упростить расчет изменения наблюдае-

```
1: PROCEDURE RandomizationStep(G; m)
2:     DO
3:         e1 = GetRandomEdge(G)
4:         e2 = GetRandomEdge(G)
5:     WHILE EdgeExists(e1.source; e2.source; G) OR EdgeExists(e1.target; e2.target; G)
6:     RemoveEdge(e1; G)
7:     RemoveEdge(e2; G)
8:     D = CalculatePropertyChange(G; e1; e2)
9:     AddEdge(e1.source; e2.source; G)
10:    AddEdge(e1.target; e2.target; G)
11:    p = CalculateProbability(D; m) . Calculated by 1
12:     IF 0 == BernoulliDistribution(p) THEN
13:         RevertChanges(G)
14:         D = 0
15:    RETURN D
```

Алгоритм 3. Шаг процесса эволюции с сохранением степеней

мой величины. Для вершин каждого из добавляемых ребер рассматриваются все общие соседние вершины. Каждая такая вершина входит в цикл длины три, который содержит добавленное ребро. Здесь важным является тот факт, что удаляемые ребра на данном этапе не рассматриваются, так как уже удалены. Для вершин каждого из удаляемых ребер также рассматриваются все общие соседние вершины – каждая такая вершина является частью цикла длины три, который уже удален. Общая величина изменения числа циклов длины три будет равна разности двух рассчитанных величин.

```
1: PROCEDURE CalculatePropertyChange(G; e1; e2)
2:     N_added = 0, N_removed = 0
3:     THREAD_1
4:         S = {v, v ∈ NEIGHBOURS(e1.source) ∩ NEIGHBOURS(e2.source)}
5:         N_added += |S|
6:     THREAD_2
7:         S = {v, v ∈ NEIGHBOURS(e1.target) ∩ NEIGHBOURS(e2.target)}
8:         N_added += |S|
9:     THREAD_3
7:         S = {v, v ∈ NEIGHBOURS(e1.source) ∩ NEIGHBOURS(e1.target)}
8:         N_removed += |S|
9:     THREAD_4
7:         S = {v, v ∈ NEIGHBOURS(e2.source) ∩ NEIGHBOURS(e2.target)}
8:         N_removed += |S|
```

Алгоритм 4. Алгоритм расчета изменения количества циклов длины три в случае сохранения степеней вершин

3.4.2. Алгоритм переключения без сохранения степеней вершин

Далее, алгоритм 5 и алгоритм 6 представляют псевдокод модифицированного (обобщенного) процесса рандомизации для случая без сохранения степеней вершин и шага процесса, соответственно. В начале, алгоритм равномерным образом из множества всех ребер выбирает случайное ребро; из множества всех пар несоединенных вершин равномерным образом выбирается случайная пара вершин. Выбранное ребро удаляется, выбранная пара соединяется ребром. Важным является равномерный выбор пары несвязанных вершин. Данный факт можно обеспечить двумя основными способами. Первый – получение множества несоединённых пар вершин на каждом шаге процесса. Ясно, что данный подход неэффективен ввиду множества накладных расчетов на каждом шаге процесса. Поэтому применяется второй подход – начальный расчёт данного значения и последующее его обновление. Накладные расходы на поддержание актуального состояния множества в данном случае малы из-за локальности производимых изменений.

Сам алгоритм сбора необходимых пар является квадратичным и поддается параллелизации

```
1: PROCEDURE Randomization(G; m)
2:     trajectory = []
3:     prop CalculateObservedProperty(G)
4:     trajectory.append(prop)
5:     pairs = FindAllUnconnectedVertexPairs(G)
6:     WHILE !Stabilized(trajectory) DO
7:         D_prop = RandomizationStep(G; m)
8:         prop += D_prop
9:         trajectory.append(prop)
10:    RETURN trajectory
```

Алгоритм 6. Модифицированный алгоритм рандомизации

при помощи директив стандарта OpenMP. Как и в случае процесса с сохранением степеней уз-

```
1: PROCEDURE RandomizationStep(G; m)
2:     e = GetRandomEdge(G)
3:     <v1; v2> = GetRandomPair(pairs)
4:     RemoveEdge(G; e)
5:     D = CalculatePropertyChange(G; e; v1; v2)
6:     AddEdge(G; v1; v2)
7:     UpdateUnconnectedPairs(pairs)
8:     p = CalculateProbability(D; m)      // Рассчитывается по формуле из главы 2
9:     IF 0 == BernoulliDistribution(p) THEN
10:        RevertChanges(G)
11:        D = 0
12:    RETURN D
```

Алгоритм 5. Шаг процесса эволюции без сохранения степеней

лов учитывается локальная природа данного переключения и рассмотрение выбранного ребра и пары вершин производится в отдельных потоках. Участок вычисления вероятности приёма шага и дальнейшие действия аналогичны предыдущему случаю.

Далее представлен алгоритм подсчета изменения количества циклов длины три при переключении связей без сохранения степеней вершин (алгоритм 7). До начала подсчета изменения рассматриваемой величины, из сети удаляется заранее выбранное ребро, но выбранная пара вершин не соединяется. Как и в первом случае этот факт позволяет упростить проводимые расчеты. Далее, параллельно выполняются две задачи. Первая задача для пары вершин удаленного ребра рассматривает всех общих соседей и находит их количество. Это число также показывает количество удаленных циклов длины три. Потом рассматривается пара выбранных для создания новой связи вершин. Как и в предыдущем случае, рассматриваются все вершины, являющиеся общими соседями этих двух вершин. Количество таких вершин есть число циклов длины три, на которое увеличится их общее количество в результате добавления новой

```
1: PROCEDURE CalculatePropertyChange(G; e; v1; v2)
2:     N_added = 0, N_removed = 0
3:     THREAD_1
4:         S = {v, v ∈ NEIGHBOURS(v1) ∩ NEIGHBOURS(v2)}
5:         N_added = |S|
6:     THREAD_2
7:         S = {v, v ∈ NEIGHBOURS(e.source) ∩ NEIGHBOURS(e.target)}
8:         N_removed = |S|
9:     RETURN N_added − N_removed
```

Алгоритм 7. Алгоритм расчета изменения количества циклов длины три в случае без сохранения степеней вершин

связи. Так как ребра, выбранного для удаления, на данном этапе уже нет, а две вершины для новой связи ещё не соединены, то пересечения между двумя рассмотренными множествами циклов длины три, количество которых было подсчитано, нет. Поэтому, результатом данной процедуры является разность между двумя рассчитанными числами.

3.5. Ключевые детали процесса реализации

Реализация системы проводилась в несколько этапов.

1) Анализ требований и возможных решений для их выполнения

   Данный этап является одним из важнейших – неправильный выбор решений или учёт не всех требований может привести к проблемам на следующих этапах. Причём насколько позже будет обнаружено несоответствие выбранного решения поставленной за-

даче, насколько больше работы придётся переделывать. Каждое из требований было отдельно проанализировано, были рассмотрены возможные подходы для решения возникающих задач, для каждого из решений были проанализированы положительные и отрицательные стороны и на основе всей имеющейся информации были приняты решения по использованию удовлетворяющих всем требованиям решений.

На данном этапе также были проанализированы доступные программные средства и выбраны наиболее оптимальные для реализации системы. Для написания ядра был выбран язык программирования C++, технология распределенных вычислений MPI, библиотека boost; для интерфейса пользователя - командный язык Bash, система визуализации данных gnuplot.

2) Проектирование архитектуры системы.

   Система спроектирована с применением модульной архитектуры – все основные функции системы распределены по отдельным модулям. Кроме того, модули, имеющие взаимосвязанные функциональности, объединены в уровни. В общем, были выделены три основных уровня системы и определены функции каждого из уровней. Основная функциональность системы была возложена на «ядро». Функции ядра включают всю базовую работу с сетями во внутреннем формате представления – загрузка сети в память, сохранение сети на диске, реализация процесса направленного развития сети, извлечение кластерной структуры сети, и т.д. Для интерфейса пользователя были зафиксированы стандартные методы взаимодействия с системой – команды, их опции с допустимыми значениями, входные и выходные форматы данных. Функции по обеспечению необходимых действий для взаимодействия интерфейса пользователя и ядра были возложены на уровень стыковки. Были определены основные потоки работы системы – поток данных и поток выполнения.

3) Реализация ключевых компонентов и создание необходимой инфраструктуры

   На данном этапе была проведена разработка инфраструктуры, что включает реализацию основных классов и методов их взаимодействия. Были спроектированы методы хранения сетей в оперативной памяти, основные методы работы с сетями, имплементированы основные алгоритмы направленного развития сетей. Были разработаны начальные версии пользовательских скриптов для работы с ядром системы и налажены механизмы работы системы на уровне стыковки.

Наличие основной функциональности и надежной инфраструктуры позволило упростить дальнейшую имплементацию системы и создать хорошую основу для последующего развития системы.

4) Тестирование базовых функций системы и налаживание алгоритмов работы

   Для проверки основных функций и алгоритмов работы системы на данном этапе были разработаны соответствующие средства автоматического тестирования и написаны базовые тесты, позволяющие избежать регрессий в ходе разработки и оптимизации системы. Кроме того, было проведено ручное тестирование основной функциональности. Были получены начальные оценки оптимальности по времени выбранных решений и сделаны соответствующие корректировки.

   Для начальной наладки кластерных вычислений использовался кластер НАН РА и суперкомпьютер ЕрФИ. Дальнейшие эксперименты проводились на кластере Института химической физики им. Н. Н. Семёнова РАН.

   Все описанные процедуры позволили выявить и исправить различные недочёты и ошибки в системе. Это привело к стабилизации системы и повышению надежности системы в целом.

5) Оптимизация различных частей для повышения производительности

   По мере разработки и тестирования системы на производительность были выявлены основные участки ядра, которые в большей степени влияют на производительность. Данные участки были детально проанализированы и были рассмотрены возможные варианты улучшения и оптимизации этих участков. Ввиду их малого объёма, каждый из вариантов был реализован и проверен на производительность. Далее были выбраны наиболее оптимальные варианты из рассмотренных. В результате производительность системы была значительно улучшена. Кроме того, были улучшены некоторые механизмы взаимодействия различных объектов системы и внутреннее устройство компонентов.

3.6. Архитектура разработанной системы

3.6.1. Высокоуровневая архитектура

При разработке системы были изучены некоторые распространенные программные среды, была проанализирована их общая архитектура. Основные идеи были учтены при разработке данной системы. Были учтены также распространенные методы решения возникающих про-

межуточных задач организации и взаимодействия компонентов программного обеспечения (шаблоны проектирования) ([85] [86]).

Для обеспечения гибкости, расширяемости и ремонтопригодности программные системы принято разделять на два основных компонента.

1) Ядро системы

   Данный компонент содержит реализации основных функциональных возможностей приложения и обеспечивает остальные компоненты системы необходимым интерфейсом для выполнения соответствующих задач.

2) Графический/командный интерфейс

   Данный компонент отвечает за взаимодействие пользователя с приложением. Все основные задачи направляются ядру для выполнения и получения результата.

Кроме двух описанных ключевых компонента, программные системы обычно содержат дополнительные уровни, которые в основном являются посредниками между ядром и интерфейсом пользователя. В функции таких уровней могут входить конвертация данных, промежуточное кэширование результатов сложных вычислений, а в случае графических приложений – наблюдение за операциями и данными ядра с целью обеспечения отображения состояния приложения в реальном времени.

Разработанная система спроектирована с учетом представленного подхода к разработке программных решений. Далее приведен рисунок общей архитектуры системы.

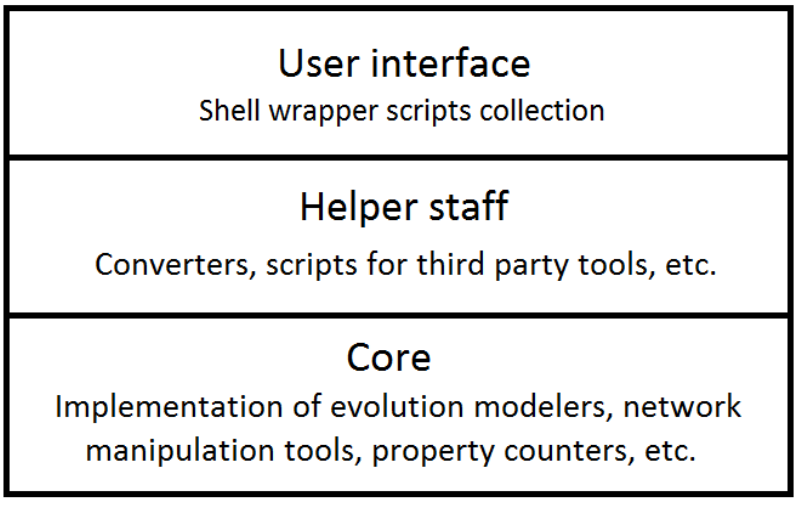


Рисунок 6. Высокоуровневая архитектура системы

В целом, как показано на рисунке, можно выделить три основных уровня системы.

1. *Интерфейс пользователя* - самый верхний и, следовательно, самый абстрактный, уровень. Данный уровень представляет набор команд, опций и форматов, при помощи ко-

торых пользователь взаимодействует с системой. Это включает описания доступных пользователю команд с набором поддерживаемых опций, форматы данных, которые воспринимаются системой как входные, абстрактные методы взаимодействия различных модулей системы (например, описание шагов, необходимых для передачи данных, полученных от генератора траекторий в визуализатор траекторий), а также форматы выходных данных. Наличие данной информации позволяет пользователю работать на данной системе и проводить различные исследования, связанные с процессом направленного развития, а также с анализом полученных либо уже имеющихся данных. Данный уровень, в основном, состоит из «оберточных» скриптов, позволяющих отделить необходимость проверки, конвертации параметров и данных от программ, реально выполняющих необходимые действия. Наличие «оберточных» скриптов придает гибкость программной среде и позволяет выполнять дополнительные действия перед запуском реальной программы, которая, в свою очередь, предполагает наличие корректных параметров и данных. Другая составляющая данного уровня – спецификация формата входных и выходных данных, позволяет состыковывать данные, полученные при помощи данной системы, с другими программными средами и наоборот – данные других систем с разработанной системой. Необходимые действия для выполнения стыковки выполняет нижний уровень.

2. *Вспомогательные средства* - средний уровень системы. Данный уровень предназначен для взаимодействия двух других уровней системы – интерфейса пользователя и «ядра». Основной функцией данного уровня является стыковка различных потоков из интерфейса пользователя в ядро и включают конвертирование данных из одного формата в другой, выполняющие вызовы сторонних библиотек и инструментов скрипты, подготовка данных для передачи от генератора траекторий к визуализатору и другие подобные вспомогательные функции. Также, в функции данного уровня входят функции обратного конвертирования – из внутреннего формата ядра в форматы, воспринимаемые интерфейсом пользователя и сторонними библиотеками. Данный уровень является гибридным, так как включает как имплементацию в виде скриптов, так и низкоуровневые программы в местах, требующих высокую производительность.
3. «Ядро» - самый низкий уровень системы. Данный уровень содержит программные имплементации основных алгоритмов работы – для моделирования процесса направленного развития, для быстрой сериализации и десериализации данных, для вычисления

спектральных свойств матриц смежности сетей, для выявления кластерной структуры сетей и т.д. Для моделирования эволюции поддерживается возможность параллельного запуска нескольких задач, в том числе и с поддержкой распределенных вычислений на основе технологии MPI и параллельных вычислений на основе пула потоков и технологии OpenMP. На данном уровне форматы входных и выходных данных зафиксированы и, как было сказано выше, поддержка других распространенных форматов оставлена на высшие уровни. Это позволяет упростить алгоритмы работы с данными и добавляет гибкости – любой формат может быть сконвертирован во внутренний формат и передан на вход программе. Поддержка новых форматов данных затрагивает только уровень вспомогательных средств и не требует изменений в ядре системы. Данный уровень реализован на языке программирования C++, что обеспечивает высокую производительность в сочетании с использованием технологий MPI/OpenMP. Однако, язык программирования C++ является компилируемым, из-за чего изменения на данном уровне требуют сборки всех зависящих от изменяемого кода модулей и может привести к специфичным проблемам.

3.6.2. Модули ядра и взаимодействие между ними

Ядро разработанной системы имеет модульную структуру. Далее представлены основные модули.

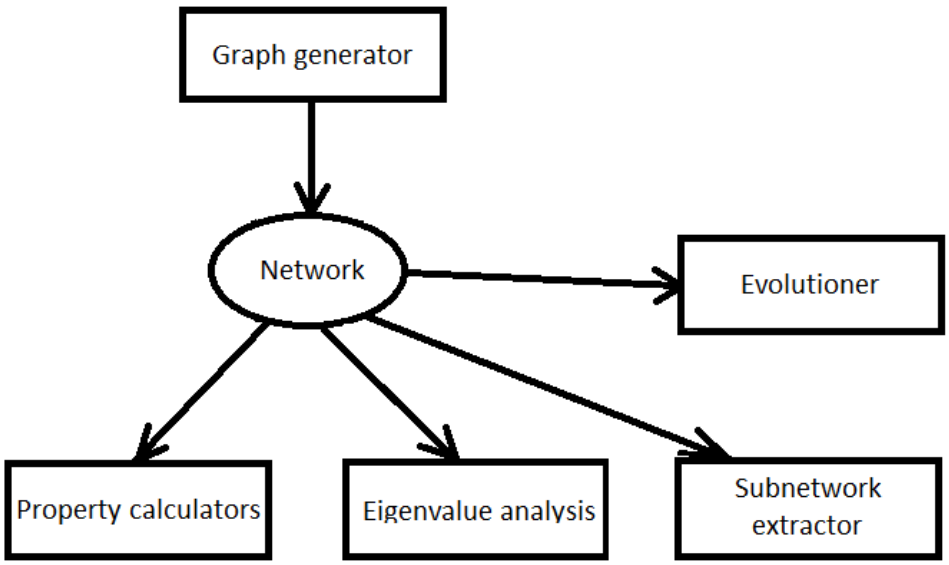


Рисунок 7. Архитектура ядра

Хранение сети в оперативной памяти

Основным модулем ядра и системы в целом является модуль хранения сети в памяти. От оптимальности операций данного модуля во многом зависит общая производительность системы. Данный модуль предоставляет основные функции работы с сетями – добавление и удаление ребер, возможность обхода всех соседей фиксированной вершины, возможность добавления и удаления вершин. Применяется метод хранения сети в виде списка смежности, т.е. для каждой вершины держится последовательность соседних вершин. При решении поставленных задач данное представления является достаточно компактным благодаря тому, что сети, рассматриваемые в процессе исследования, являются разреженными и, кроме того, издержки хранения служебных данных минимальны. Кроме того, для поддержки вычислений на плотных сетях, предусмотрена возможность смены метода хранения данных с целью оптимизации расхода памяти, а именно – вместо списка смежности хранится побитовое представление тех элементов матрицы смежности, которые лежат над диагональю. Данное представление является однозначным ввиду симметричности матрицы смежности неориентированных графов. Поддерживается сериализация и десериализация сетей в фиксированный внутренний формат представления. Ядро поддерживает возможность генерации сетей модели Эрдёша-Реньи, а уровень вспомогательных средств добавляет возможности конвертации внутреннего формата представления в общепринятый формат CSV и обратно.

Расчет основных характеристик сетей

Для подсчета различных свойств сетей реализован соответствующий подмодуль. Данный подмодуль предлагает широкий спектр рассчитываемых величин:

- Характеристики траекторий (среднее значение, среднеквадратичное отклонение)
- Среднюю длину пути, диаметр
- Коэффициенты кластеризации (локальный и средний)
- Коэффициенты центральности (betweenness centrality, closeness centrality)

Основным форматом данных, используемом для выдачи данных, является формат CSV. На уровне вспомогательных средств предусмотрены методы конвертирования данных в некоторые форматы, удобные для анализа данных с помощью специализированных программных пакетов.

Извлечение кластерной структуры

Важной составляющей структурного анализа является анализ отдельных кластеров, составляющих сеть. Данную задачу выполняет соответствующий модуль системы. На вход данный

модуль получает сеть во внутреннем формате представления и выдает необходимую структуру в том же формате, благодаря чему весь предлагаемый спектр характеристик может быть непосредственно вычислен и для полученных подграфов. Поддерживаются различные варианты извлечения подструктур сети:

- Извлечение подграфов, заданных посредством списка вершин
- Удаление списка заданных вершин
- Выявление кластерной архитектуры сети

  Для данной цели используется распространенный и эффективный алгоритм Louvain [66]. На выходе выдается информация о составляющих сеть кластерах.

Генератор траекторий

Средства генерации траекторий на основе заданных параметров выделены в отдельный модуль, который называется генератором траекторий. Данный модуль содержит имплементацию алгоритмов процесса направленного развития сетей и средства взаимодействия между единицами выполнения. Для поддержки стандарта MPI выбрана интерфейсная оболочка поверх стандарта, позволяющая не только абстракцию на объектном уровне, но и независимость от библиотеки имплементации стандарта. Стоит отметить, что последнее производится не в ущерб производительности. Использование интерфейсной оболочки повышает гибкость и позволяет переносить систему на множество поддерживаемых имплементаций библиотеки MPI.

Анализатор спектра

Данный модуль содержит необходимые средства для получения спектра собственных значений входных сетей, заданных во внутреннем формате. На выходе данный модуль выдает отсортированную последовательность собственных значений.

3.6.3. Структурные единицы ядра

На диаграмме 1 показаны ключевые компоненты системы и взаимодействие двух работающих экземпляров процесса эволюции. Основной структурной единицей ядра является класс приложения. Данный объект наделен следующими функциями:

1) Обработка входных аргументов, проверка правильности заданных значений и их типов, подстановка значений по умолчанию.
2) Считывание входных данных, таких как сеть, последовательность значений параметра μ, из соответствующих потоков.

3) Анализ среды выполнения с целью выявления оптимальных параметров распределения задач.
4) Создание задач в соответствии с имеющимся параметрами, распределение данных и постановка на выполнение.
5) Методы взаимодействия между выполняющимися процессами.
6) Получение результатов от каждой задачи и сброс полученных данных на жесткий диск.

Вся работа со средой MPI возложена на данный класс. Остальные единицы не должны иметь непосредственного доступа к функциям MPI, т.е. класс приложения является единственным объектом, знающим о наличии среды MPI и знающим как ею пользоваться. Подобное ограничение позволяет реализовать следующие важные идеи:

1) Логическое разделение единиц по выполняемым задачам. Основная задача класса приложения является работа со средой выполнения (будь то среда MPI, или же окружающая среда выполнения, значения не имеет) и остальные единицы, при необходимости, должны обращаться к данному классу.
2) При наличии ошибок, связанных с передачей данных, запуском задач или же распределением задач, круг поиска ограничивается только классом приложения.
3) Кроме того, остальные единицы не имеют зависимостей от библиотек, используемых классом приложения.

Основной единицей выполнения процесса направленного развития является класс задачи (base_task на диаграмме 1). Данный класс является интерфейсом и предоставляет только абстрактные методы запуска и получения результатов. Реальные алгоритмы для получения необхо-

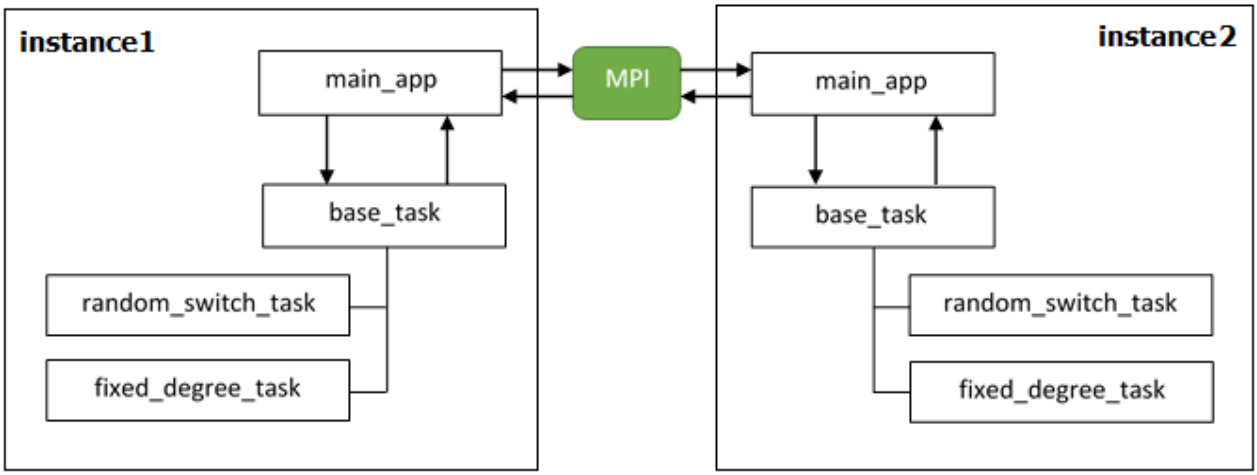


Диаграмма 1. Взаимодействие компонентов системы

димых результатов находятся в классах, наследуемых данный интерфейс. Это позволяет с легкостью добавлять новые алгоритмы процесса направленного развития сетей без изменения

других единиц. В разработанной системе реализованы два основных класса, которые обеспечивают два варианта переключения связей – с сохранением степеней вершин и без сохранения степеней (random_switch_task и random_switch соответственно на рисунке диаграмме 1).

3.6.4. Поток выполнения и поток данных

Далее будут представлены два основных потока ядра системы – поток выполнения и поток данных. Данные потоки позволяет лучше понять принципы работы системы и иметь представление о выполняемых шагах при генерации результатов.

Поток выполнения

Последовательность шагов выполнения следующая:

1) Запуск приложения

   Данный шаг является началом работы системы. Пользователь подготавливает необходимые данные, выбирает значения параметров и передаёт все данные на вход системе.

2) Обработка входных аргументов, проверка правильности заданных значений и их типов, подстановка значений по умолчанию.

   Данный шаг отфильтровывает все некорректные вызовы команд, а также проверяет правильность переданных данных, существование и доступ к входным файлам. Также, для параметров, имеющих значения по умолчанию, но не определенных пользователем, подставляются соответствующие значения. При наличии ошибок или неправильных данных, исполнение прерывается и выдается сообщение об ошибке. В противном случае, управление передается на следующий шаг.

3) Считывание входных данных - сеть, последовательность значений параметра μ, значения аргументов.

   На данном шаге из файлов и стандартного потока ввода считываются все необходимые параметры и данные и помещаются в соответствующие структуры в памяти. Управление передается следующему шагу.

4) Анализ среды выполнения с целью выявления оптимальных параметров распределения задач.

   На данном шаге производится анализ заданных параметров и выявляется оптимальное число задач для каждого из запущенных процессов. Входные данные разделяются на соответствующие части для каждого процесса и производится переход на следующий шаг.

5) Создание задач в соответствии с имеющимся параметрами, распределение данных и

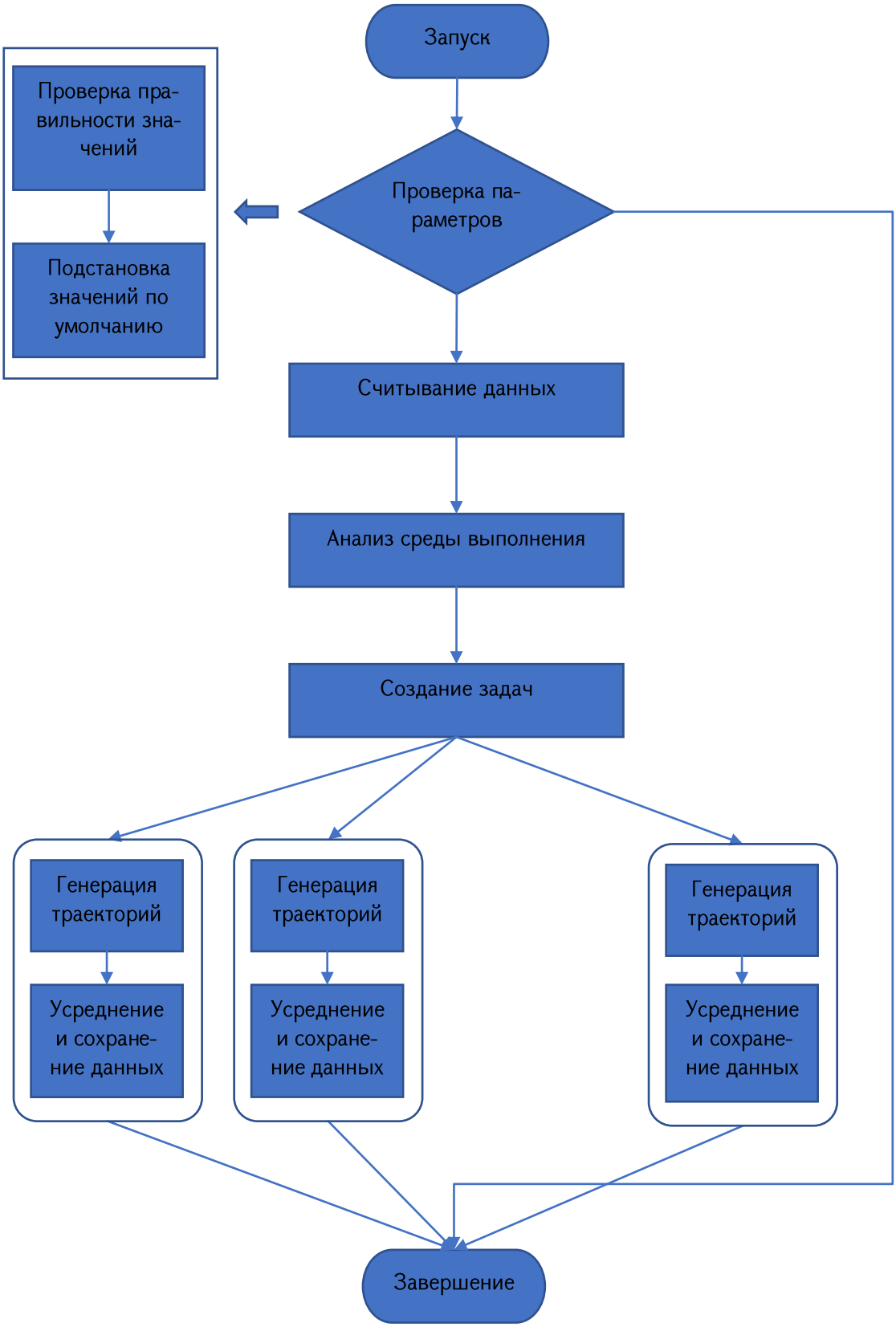


Диаграмма 2. Поток выполнения

постановка на выполнение.

Подготовленные для каждого процесса данные передаются (условно, в реальности все процессы загружают данные полностью, однако после распределения данных, у каждого процесса остаётся только необходимая часть) на вход для создания соответствующей задачи, которая далее ставится на выполнение.

6) Выполнение запущенных задач

   На данном этапе производится генерация траектории с сохранением сетей на заданных участках траекторий. Также, при задании соответствующих параметров, генерируются несколько траекторий с одинаковыми параметрами. После завершения задачи, производится переход на следующий шаг процесса.

7) Усреднение данных по нескольким траекториям с одинаковыми параметрами запуска.

   Данные множества траекторий, запущенных с идентичными значениями параметров, усредняются и поступают на вход следующему шагу процесса.

8) Получение результатов и сброс данных на жесткий диск.

   Все полученные траектории записываются на диск в заранее определенный каталог.

9) Завершение выполнения задачи.

Диаграмма 2 показывает данную последовательность.

Поток данных

Основными данными в течении процесса являются эволюционная траектория и состояние сети в различные моменты времени. Поток данных имеет следующую последовательность действий:

1) Запуск приложения
2) Считывание данных сетей с диска

   На данном шаге с диска считывается входная сеть и сохраняется в соответствующей структуре в памяти.

3) Распределение данных по процессам

   После определения задач для каждого процесса, каждый из них получает все необходимые данные и запускает процесс эволюции.

4) Получение значений рассматриваемого свойства

   На каждом шаге процесса (или же на некотором заранее определенном интервале) полученное значение рассматриваемого свойства (циклы длины три) сохраняется в памяти для последующей обработки.

5) Получение и сохранение на диске сетей на заданных участках процесса

В заданных точках траектории получается состояние сети и записывается на диск.

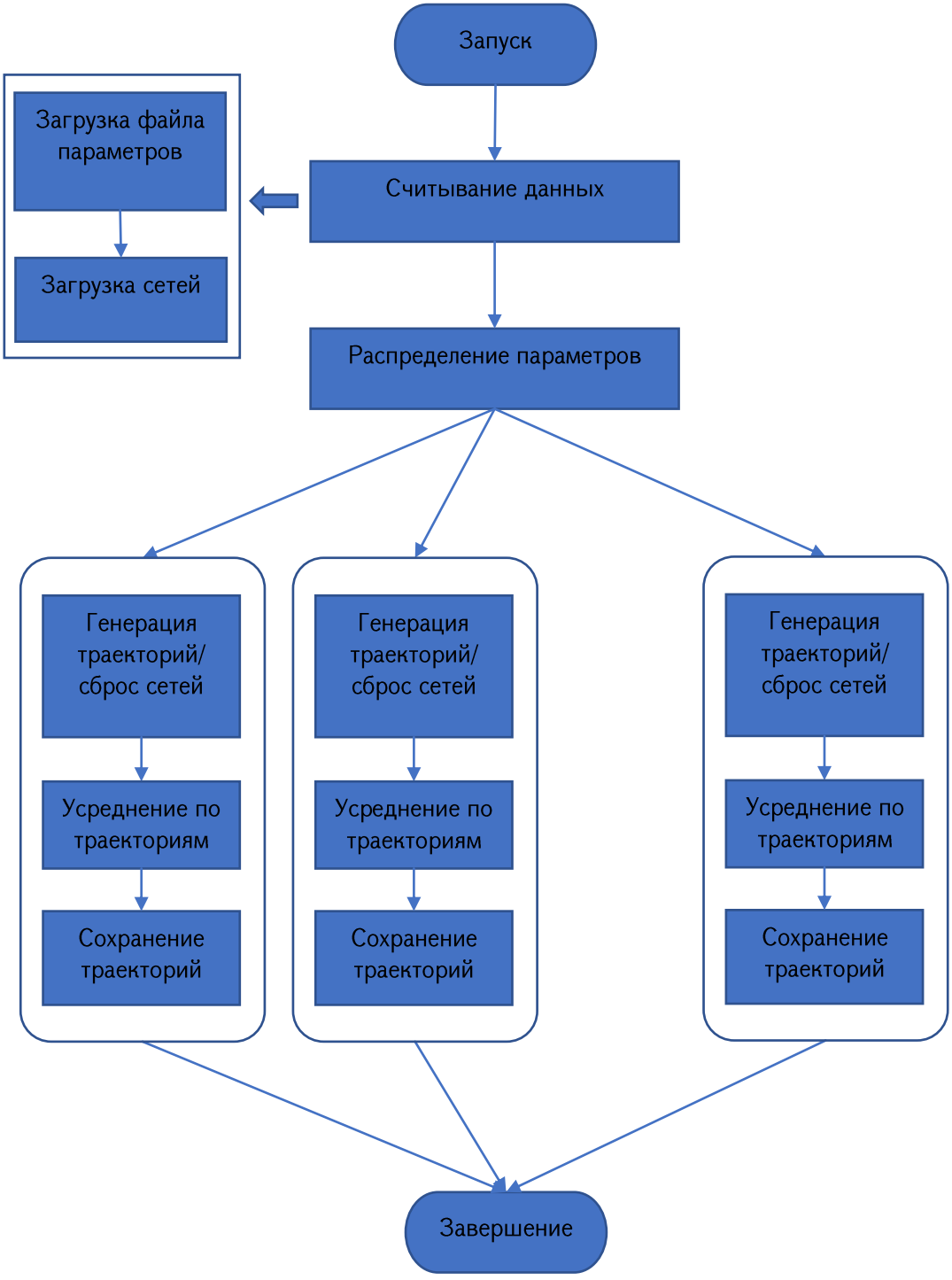


Диаграмма 3. Поток данных

6) Усреднение данных по нескольким траекториям

   При использовании опции усреднения по нескольким траекториям, данные соответствующих траекторий усредняются.

7) Сохранение траектории на диске

   Полученные данные траектории сохраняются на диске.

8) Завершение работы

На диаграмме 3 приведена описанная последовательность шагов.

3.6.5. Утилитарные компоненты

В систему включены утилитарные компоненты, которые предназначены для выполнения, упрощения и ускорения различных участков проводимых исследований. Данные компоненты используют сторонние библиотеки и инструменты для предоставления следующих функций:

1. Визуализация траекторий и сетей.
2. Получение динамической картины изменения структуры сетей.
3. Визуализация матриц смежности сетей.
4. Получение видеоматериалов для структурных изменений матрицы смежности сетей.
5. Визуализация спектра и спектральной плотности.

3.7. Характеристики системы

3.7.1. Временные данные

Таблица 1 представляет данные по производительности системы для процесса направленного развития при различных параметрах генерации входных сетей. Испытания производились на случайных сетях модели Эрдёша-Реньи. В таблице приведены данные для генерации траекторий длиной $5x10^6$ для двух случаев – с сохранением степеней и без сохранения степеней вершин. Приведена зависимость времени выполнения задачи в зависимости от размера и плотности сети.

N	P	Время, t_1	Время, t_2
64	0.1	0.023	0.012
	0.3	0.025	0.014
	0.5	0.031	0.016
	0.7	0.05	0.018
	0.9	0.053	0.02
256	0.1	0.05	0.04
	0.3	0.19	0.12
	0.5	0.37	0.24
	0.7	0.69	0.34
	0.9	1.39	0.39
1024	0.1	0.92	0.84
	0.3	4.7	2.5
	0.5	11.75	6.6
	0.7	29.2	12.7
	0.9	59.2	17.4

Таблица 1. Временные характеристики системы

Данные приведены в часах.

Далее приведен график зависимости времени выполнения в зависимости от параметров входных сетей и метода переключения связей (рис. 8, применена логарифмическая шкала по обоим осям).

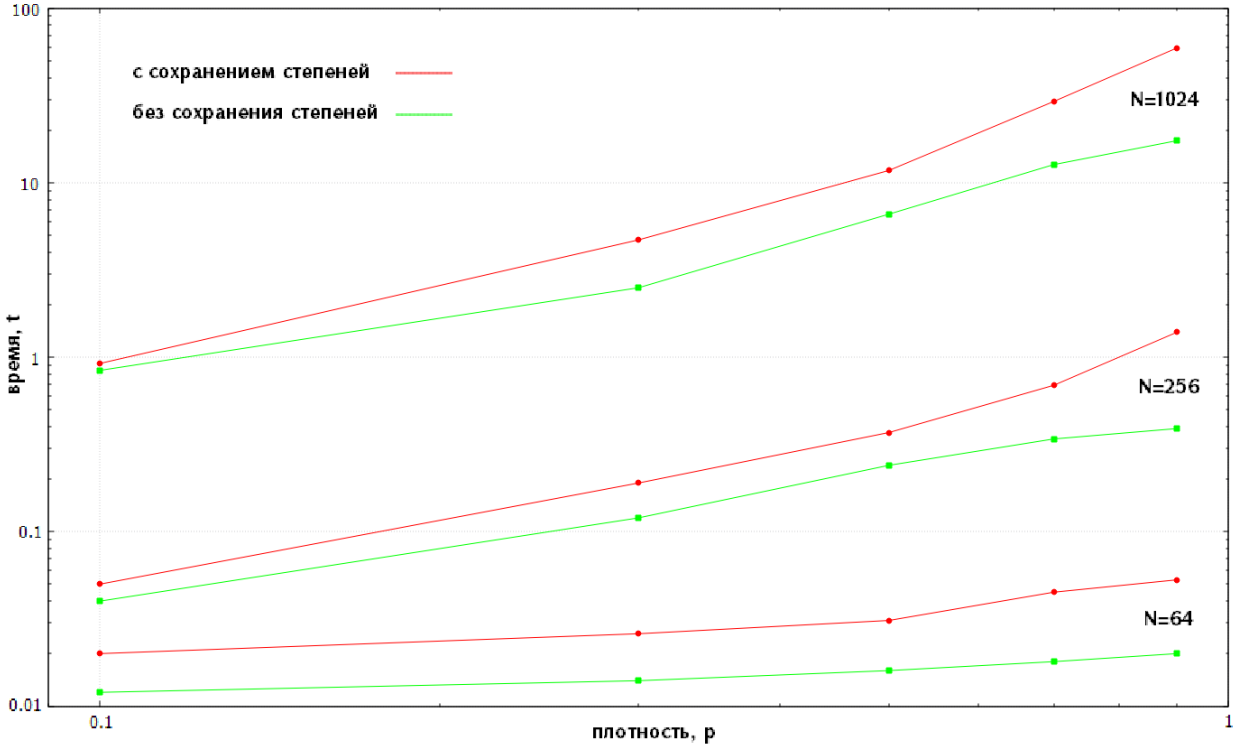


Рисунок 8. График зависимости времени выполнения от входных параметров

Как показывает рисунок, время выполнения процесса с сохранением степеней вершин больше чем время выполнения его аналога без сохранения степеней. Кроме того, видно, что время выполнения с увеличением плотности сети p в первом случае растет быстрее, чем время выполнения во втором случае.

3.7.2. Особенности системы

Разработанная система предоставляет необходимые средства для разностороннего анализа процесса направленного развития сетей и происходящих в ходе данной динамики явлений. Все поставленные требования по функциональности системы выполнены. Система разработана с учетом возможности расширения на различных уровнях. Далее представлены основные предоставляемые системой функции.

1) **Генерация случайных сетей**

   Поддерживается генерация сетей и ансамблей сетей модели Эрдёша-Реньи по заданным параметрам.

2) **Получение эволюционных траекторий**

Система предоставляет возможность получения траекторий, достигающих стабилизации. Поддерживаются два варианта переключения связей – с сохранением степеней и без сохранения степеней вершин. Получение производится с применением параллельных и распределенных технологий, благодаря чему возможна одновременная генерация множества траекторий. Поддерживается визуализация траекторий, получение сетей в заданных точках траектории, усреднение результатов по нескольким траекториям.

3) **Расчет основных свойств сетей**

Для анализа свойств сетей система предоставляет возможности получения основных свойств сетей. Поддерживается расчет следующих свойств:

- среднее значение и среднеквадратичное отклонение по траекториям
- средняя длина пути и диаметр сетей
- распределение степеней вершин
- локальный и средний коэффициент кластеризации
- коэффициенты центральности

4) **Получение структурной картины сетей**

Система предоставляет необходимые средства для получения графического представления структуры сетей. Поддерживается возможность выявления и отделения кластеров из структуры сетей и их последующий анализ.

5) **Анализ матриц смежности сетей**

Для анализа матриц смежности сетей система предоставляет возможности их визуализации и получения динамики изменения структуры. Предоставляются средства для получения спектральных свойств матриц, таких как:

- спектр
- спектральная плотность
- распределение расстояний между собственными значениями

Поддерживаются различные варианты визуализации – в виде множества точек, гистограмм и тепловых карт.

6) **Получение фото- и видеоматериалов динамик развития**

Система предоставляет возможность получения видеоматериалов для наглядного представления динамики развития структуры сетей и структуры матриц. Также поддержива-

ется возможность получения фотоматериалов для динамики развития спектральной плотности.

7) **Импорт/экспорт сетей в стандартный формат CSV**

   Поддерживается возможность работы со стандартным форматом представления данных CSV. Все выходные данные сохраняются в данном формате. Данный формат поддерживается многими стандартными пакетами, и его поддержка позволяет передачу данных между разработанной системой и пакетами.

Ниже представлены основные особенности системы:

1) **Удобный интерфейс пользователя**

   Система разрабатывалась с учетом удобства работы с ней. Разработанные команды предоставляют краткое описание выполняемой работы и доступных для использования опций. В процессе работы выдаётся состояние запущенного процесса.

2) **Расширяемость**

   В системе предусмотрена возможность расширения. Расширение предусматривает возможность добавления новых методов переключения связей на каждом шаге процесса, а также новых свойств, которые необходимо максимизировать в ходе процесса развития.

3) **Кроссплатформенность**

   Благодаря использованию соответствующих техник и технологий основная часть системы является кроссплатформенной. Вся работа со средой выполнения и с платформой оставлена на сторонние библиотеки, которые доступны на всех основных платформах.

ГЛАВА 4: ЭКСПЕРИМЕНТАЛЬНЫЕ РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ

В данной главе представлены экспериментальные результаты исследования процесса направленного развития сетей при двух рассмотренных вариантах переключения связей на каждом шаге процесса. Раздел 4.1 описывает этапы проведенных исследований. Раздел 4.2 описывает основные результаты проведенных исследований – представлены эволюционные траектории, характеристики сетей, показана картина структурообразования в ходе процесса, а также представлен анализ структуры матриц смежности рассмотренных сетей и их спектральная плотность.

4.1. Этапы исследования

Проведенные исследования можно разделить на несколько основных этапа. Далее приведено последовательное описание всех этапов. Детальное описание соответствующих результатов будет представлено позднее.

4.1.1. Получение и исследование траекторий для сетей с различными параметрами

Изучалась модель Эрдёша-Реньи при различных параметрах N и p. Были проделаны следующие шаги:

1) *Нахождение критического порога* μ.

   Данный этап исследования отталкивается от работы [43], где было показано существование критического значения $\mu = \mu_c$, при котором равновесное значение $T(G)$ увеличивается резко.

   Целью данного этапа является определение порогового значения параметра μ. Критический порог – это минимальное значение параметра μ, при котором наблюдается резкий рост количества циклов длины три в сети. Кроме того, что это значение должно быть минимальным, необходимо чтобы для всех значений параметра, больших найденного, тоже наблюдался резкий рост наблюдаемого свойства. Данное требование отображает факт нестабильности системы вблизи критической точки.

   Изначально, производится проход большим шагом по широкому интервалу параметра μ с целью определения «грубого» значения критического порога. Для каждого значения

параметра генерируются траектории и производится начальный анализ. В первую очередь проверяется наличия скачка в количестве циклов длины три. Если подъёма не наблюдается, рассматривается возможность наличия скачка на более длиной траектории и для этого увеличивается количество производимых шагов. Если скачка опять не наблюдается, начальный интервал заменяется новым и процесс генерации повторяется. Имеет смысл рассматривать только такой интервал параметра μ, при значениях из которого вероятность принятия шага строго меньше единицы, потому что для больших значений μ рассматривать траекторию не имеет смысла – всегда принимаются только шаги, увеличивающие количество треугольников и идёт резкий подъём с самого начала процесса. Если подъёма не наблюдается, последующие этапы не рассматриваются. В случае если на некотором значении параметра наблюдается скачок, происходит переход ко второму этапу исследования.

Кроме определения порогового значения, на данном этапе также определяется порядок количества шагов, необходимых для наступления стабилизации, а также шаг значений параметра μ для генерации траектории.

2) *Построение траекторий вблизи критического порога.*

На основе начального исследования производится генерация траектории с достаточно мелким шагом для более точного выявления порога. После определения порогового значения с необходимой точностью, берется некоторый малый промежуток вокруг этого значения так, чтобы он включал как значения, где наблюдается скачок, так и значения, на которых скачок не наблюдается. Это необходимо для исследования процесса в малой окрестности порогового значения. На полученном интервале значений с равномерным шагом выбираются значения для точного определения точки скачка. Если шаг выбрать маленьким, после скачка возможно появление значений, при которых скачка не наблюдается – опять же из-за больших флуктуаций вокруг критической точки. Наоборот, если выбрать шаг большим, точность определения порогового значения будет малой. Для каждого из выбранных значений генерируется траектория с учетом информации, полученной на первом этапе. Количество шагов выбирается таким образом, чтобы возможно было наблюдение стабилизации на промежутке, достаточном для определения свойств сети на этом участке. Также количество шагов должно быть достаточным, чтобы возможно было судить о наличие больше, чем одного скачка или достаточно удаленного от

начала траектории скачка. На основе полученных траекторий производится следующий этап исследования.

3) *Анализ стабильных хвостов траектории.*

Производится анализ полученных траекторий для выяснения изменений в свойствах и строении сети.

Графики траекторий дают начальное представление о пороговом значении, высоте и длине скачка, наличии одного или нескольких дополнительных скачков, а также о стабилизационном участке. Наличие скачка наглядно демонстрируется среднеквадратическим отклонением и средним значением количества циклов длины три, вычисленными по участку стабилизации. Далее представлены формулы, используемые для расчетов.

a. Среднее по участку стабилизации вычисляется следующим образом:

$$average = \sum_{i \in stable} \frac{n_i}{count} \tag{19}$$

где $average$ – среднее по участку стабилизации, n_i – число циклов длины три на каждом шаге траектории, $stable$ – участок стабилизации, $count$ – длина участка стабилизации.

b. Среднеквадратическое отклонение подсчитывается стандартной формулой:

$$\sigma = \sqrt{\frac{\sum_{stable}(\langle n \rangle - n_i)^2}{count}} \tag{20}$$

где σ – среднеквадратическое отклонение, $\langle n \rangle$ – среднее число циклов длины три по всей траектории.

4) *Анализ свойств сетей*

Для начального понимания происходящих в сети изменений анализируются различные свойства сетей – коэффициент кластеризации, среднее расстояние между вершинами, диаметр.

5) *Получение результатов для различных значений параметров.*

На данном шаге описанный в предыдущих пунктах процесс нахождения критического порога, соответствующих траекторий и свойств производится на сетях различных размеров и плотностей. Также получается зависимость критического порога от параметров сети.

Данный этап позволяет наблюдать фазовый переход при рассматриваемой динамике и получить начальное представление о наблюдаемом явлении.

4.1.2. Исследование структуры и структурных изменений в наблюдаемых сетях

Первым шагом для исследования наблюдаемого явления является рассмотрение коэффициента кластеризации сети. Рассматривается динамика его изменения по ходу процесса развития. Также рассматриваются другие основные характеристики сетей – диаметр, среднее расстояние между вершинами, коэффициенты центральности. Далее, исследование продолжается с помощью структурного анализа финальных сетей. Для этого используются алгоритмы визуализации сетей ([61] [87] [88]), и, в частности, применяется силовой алгоритм Фрухтермана-Рейнгольда [63]. Кроме визуализации, используются алгоритмы обнаружения сообществ (community detection, [65] [66]) для выделения кластеров из рассматриваемых сетей и их последующего анализа. Следующим шагом является аналогичное исследование промежуточных сетей и получение общей картины динамики структурных изменений по ходу процесса направленного развития.

4.1.3. Исследование матриц смежности сетей

Целью данного этапа является анализ матриц смежности исследуемых сетей. Для этого вычисляются спектры матриц смежности финальных и промежуточных сетей. Для анализа спектральной плотности изучаются объединенные спектры для матриц смежности ансамбля сетей. Применяются соответствующие методы визуализации и, кроме того, техника визуализации самих матриц смежности в специальной форме, проливающей свет на структуру сетей. Кроме того, на данном этапе изучается динамика изменения структуры матриц смежности сетей в ходе процесса направленного развития.

4.2. Результаты моделирования

Данный раздел описывает основные результаты, полученные в ходе исследований при помощи разработанной программной среды. Представлены сравнительные картины двух рассмотренных случаев переключения связей в ходе процесса направленного развития сетей ([2]). Последовательно описаны полученные траектории, основные характеристики сетей, приведены сравнительные графики рассматриваемого свойства – количества циклов длины три, и других свойств сети, представлена структурная картина самого процесса, матриц смежности и спектральной плотности матриц смежности сетей.

4.2.1. Траектории, основные характеристики

Далее приводится сравнительный анализ эволюционных траекторий, полученных из одних и тех же сетей, но с различными значениями параметра μ. Приводится сравнение двух вариан-

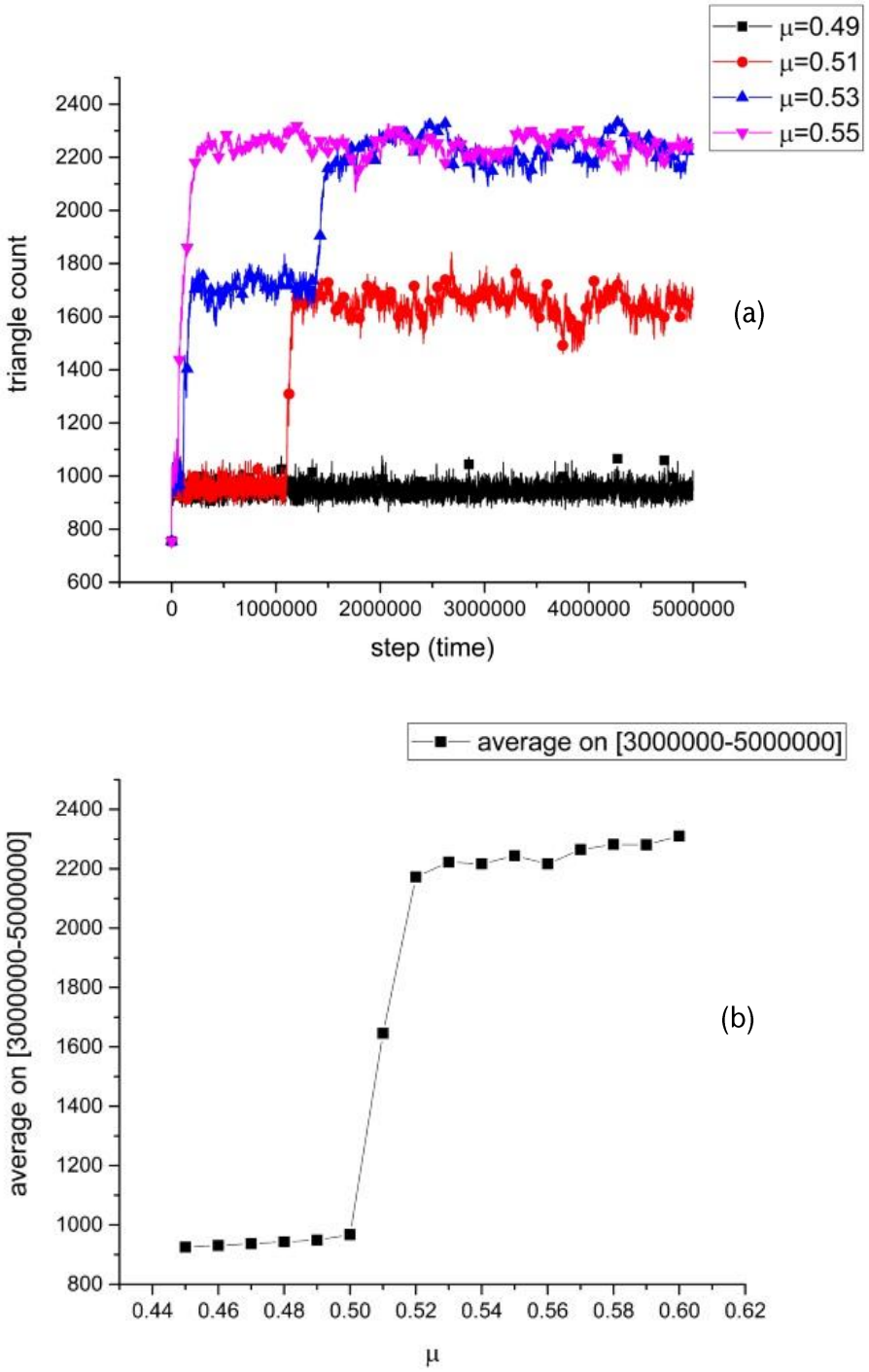


Рисунок 9. Эволюция случайной сети (N=64, p=0.3) с сохранением степеней вершин. (a) – Траектории для μ=0.49, μ=0.51, μ=0.53, μ=0.55. Переход наблюдается при μ=0.51 и μ=0.53. (b) – Равновесное значение T(G) как функции от μ.

тов процесса развития – с сохранением степеней и без сохранения степеней. Изначальное моделирование проводилось на сетях размеров 64, 128, 256 и 512. Анализировались топологические свойства и структура сетей.

Компьютерное моделирование показало, что при значениях, больших критического порога – $\mu \geq \mu_c$, эволюция с сохранением степеней приводит к последовательному образованию множества плотных кластеров. Образование кластеров сопровождается увеличением среднего коэффициента кластеризации и среднего расстояния между узлами.

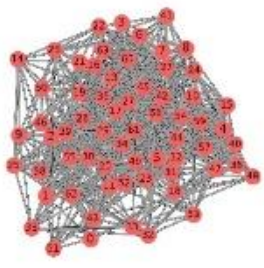
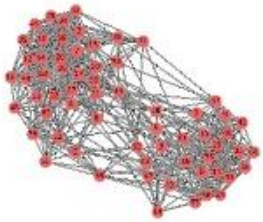
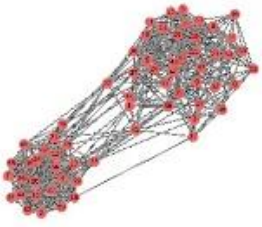
Далее будут показаны типичные результаты разделения изначально однородной сети в множество кластеров на примере сети размером 64. Такие же результаты наблюдаются и для сетей большего размера (128, 256 и 512 узлов). Для сравнения приведены структуры сетей, образовывающиеся при эволюции без топологических ограничений – т.е. без сохранения степеней узлов. Начальные сети были сгенерированы случайным образом при вероятности связей $p = 0.3$, что намного выше порога перколяции и порога связности графа.

Эволюция с сохранением степеней узлов

На рис. 9(a) показаны эволюционные траектории, полученные из одной и той же начальной сети G при различных значениях параметра μ ниже и выше критической точки μ_c. Внимания заслуживает тот факт, что некоторые траектории обладают ступенчатыми переходами. Рис. 9(b) показывает равновесное значение $T(G)$ как функцию от μ. Здесь ясно виден скачок значения $T(G)$ вблизи $\mu = 0.5$.

Траектории показывают, что при значениях параметра, больших критического порога, например, при $\mu = 0.53$, происходит ряд переходов, которые сопровождаются увеличением количества

циклов длины три. Данное явление объясняется структурным анализом сетей, сгенерированных по ходу процесса эволюции (таблица 1).

		
#1	#2	#3

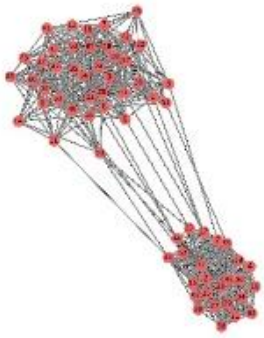
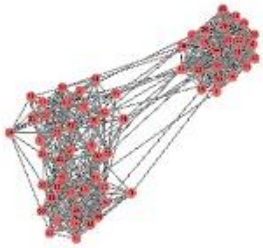
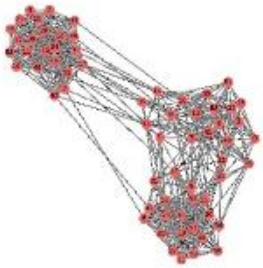
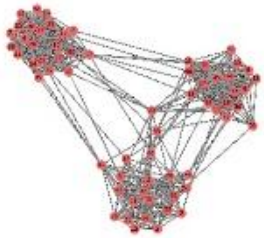
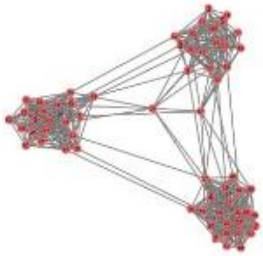
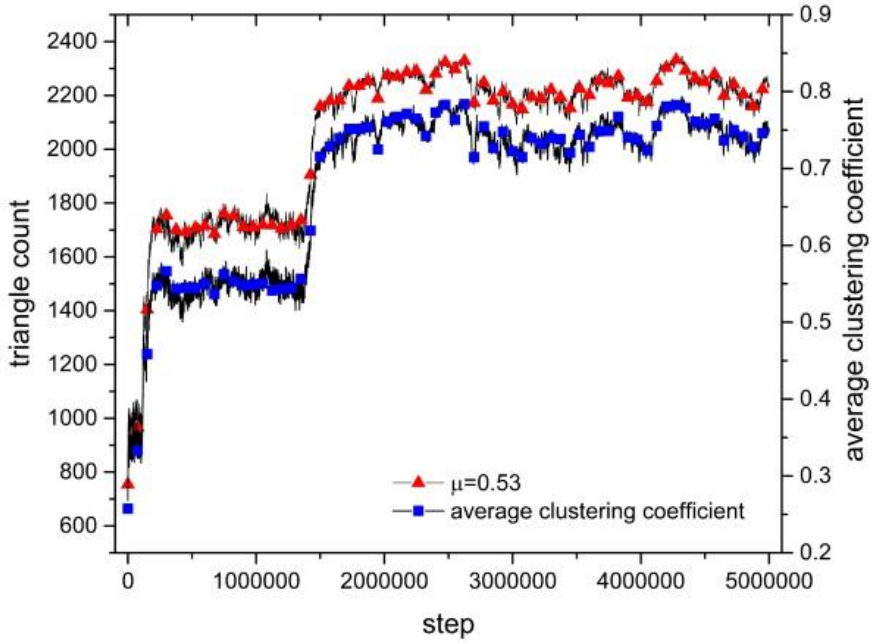
 #4	 #5	 #6
 #7	 #8	

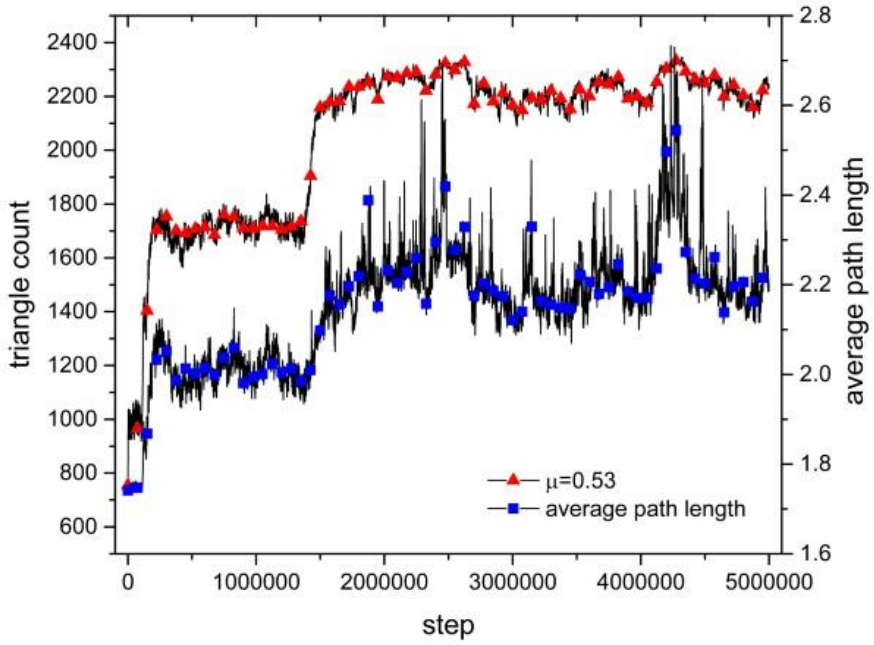
Таблица 2. Типичная картина развития сети при сохранении степеней вершин

Структуризация сети над критическим порогом обусловлена разделением сети на плотные связные компоненты. На проиллюстрированном примере, сеть сначала разделяется на два кластера, один из которых меньше, но плотнее другого, и содержит узлы с большими степенями. На следующем этапе эволюции, плотный кластер почти не подвергается каким-либо структурным изменениям, а больший второй кластер расщепляется на два меньших кластера. Дальнейшее развитие к образованию новых кластеров не приводит. На последнем рисунке (рисунок #8) показана финальная структура рассмотренной сети – три маленьких, плотных кластера связанные между собой гораздо (на порядок) слабее, чем внутри кластеров.

Кроме эволюционных траекторий были изучены изменения в топологических характеристиках сети по ходу образования кластеров. Далее представлены данные, соответствующие траектории, для которой структурная картина представлена в таблице 2. Рис.10 представляет сравнение изменения количества циклов длины три по времени и соответствующие изменения среднего расстояния между вершинами и коэффициента кластеризации сети. Ясно видно, что обе приведенные характеристики сети коррелируют с изменением количества циклов длины три в сети – все три величины ведут себя синхронно.



(a)



(b)

Рисунок 10. Сравнение различных топологических свойств в случае эволюции с сохранением степеней. Верхние кривые соответствуют количеству циклов длины три; нижние кривые соответствуют среднему коэффициенту кластеризации (a) и среднего расстояния между вершинами (b), соответственно.

Эволюция без сохранения степеней узлов

Идентичные эксперименты были проведены для случая эволюции без сохранения степеней. Анализ результатов показывает совсем иную картину, нежели случай с сохранением степеней.

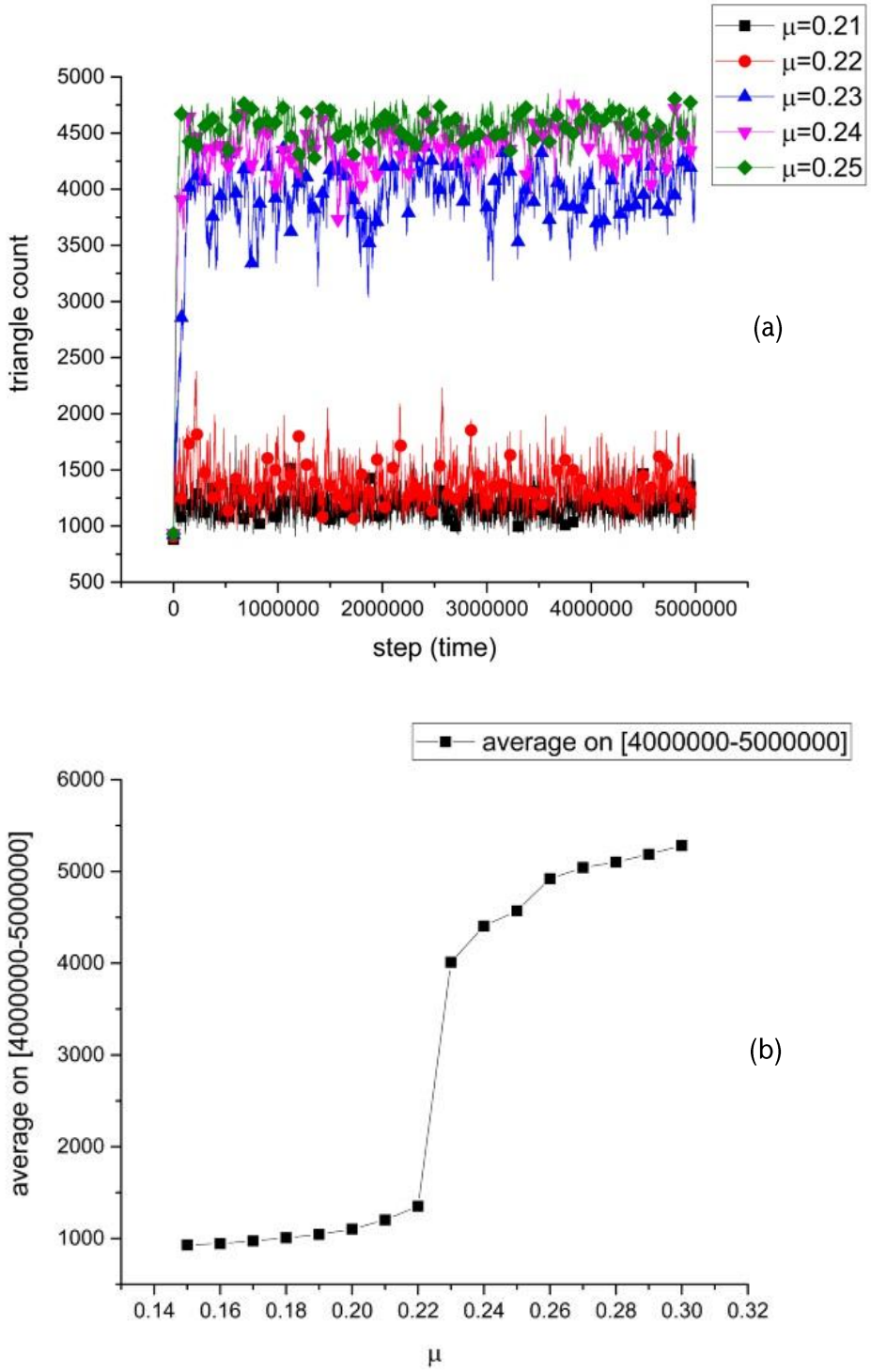
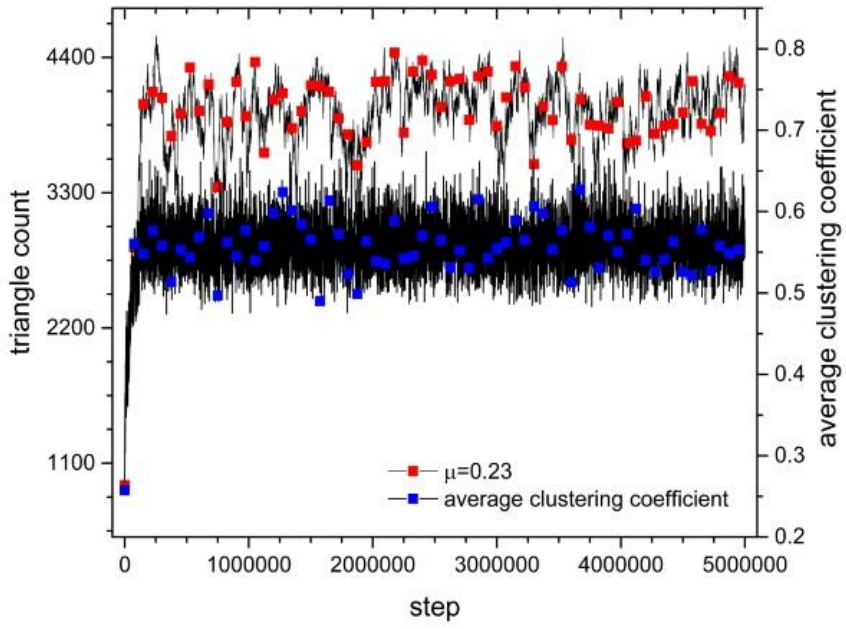
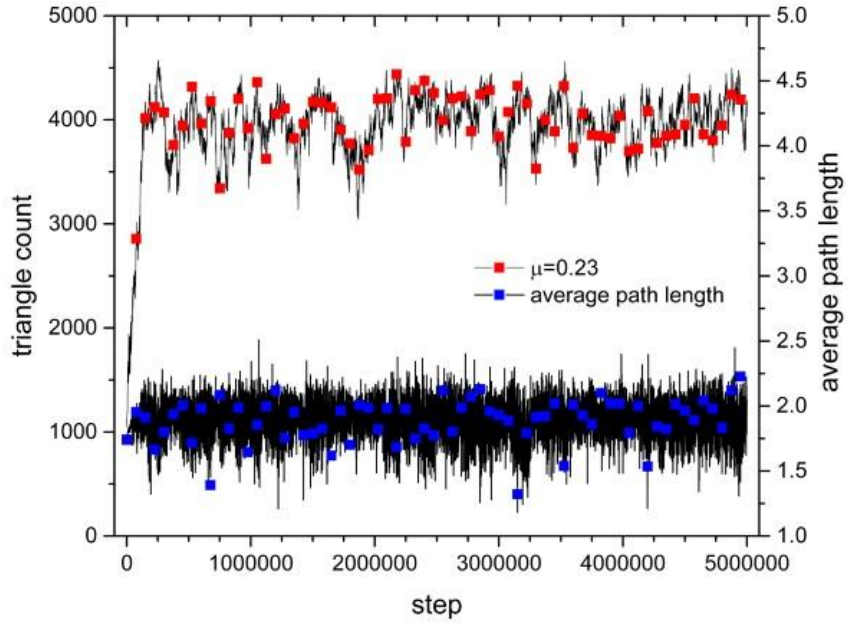


Рисунок 11. Сравнение различных топологических свойств в случае эволюции без сохранения степеней. Верхние кривые соответствуют количеству циклов длины три; нижние кривые соответствуют среднему коэффициенту кластеризации (a) и среднего расстояния между вершинами (b), соответственно.

На рис. 11(a) приведены траектории для той же сети, для которой были приведены результаты процесса развития с сохранением степеней вершин. Опять же, при значениях параметра μ выше критического значения наблюдается скачок количества циклов длины три, однако в данном случае ступенчатости не наблюдается. Стоит отметить, что ввиду отсутствия тополо-



(a)



(b)

Рисунок 12. Эволюция случайной сети (N=64,p=0.3) в случае без сохранения степеней вершин. (a) – Траектории для μ=0.21, μ=0.22, μ=0.23, μ=0.24 и μ=0.25. (b) – Равновесное значение T(G) как функции от μ. Критическая точка - μ=0.23.

гических ограничений в ходе процесса, количество циклов длины три в стабильном состоянии гораздо больше, чем в аналогичном случае с сохранением степеней. Кроме того, критический порог параметра μ, наоборот, гораздо меньше. Рис. 11(b) показывает скачок при рассмотрении среднего значения количества циклов длины три на стабильных хвостах траекторий.

Отсутствие ступенчатости объясняется отсутствием кластерной структуры в финальных сетях. В таблице 3 приведены характерные картинки развития структуры сети в ходе процесса без сохранения степеней вершин. Из рисунков видно, что начальная сеть постепенно стягивается в один плотный кластер, образуя «пар» вокруг себя в виде небольшого числа слабо связанных с кластером вершин.

На рис. 12 приведены сравнительные графики среднего коэффициента кластеризации с количеством циклов длины три и средней длины пути с количеством циклов длины три. В отличие от первого случая, где между представленные величины вели себя синхронно, в данном случае корреляции не наблюдается. Далее, средняя длина пути не изменяется в данном случае (если отбросить флуктуации и рассматривать усредненное значение) и средний коэффициент кластеризации ниже, чем в первом случае.

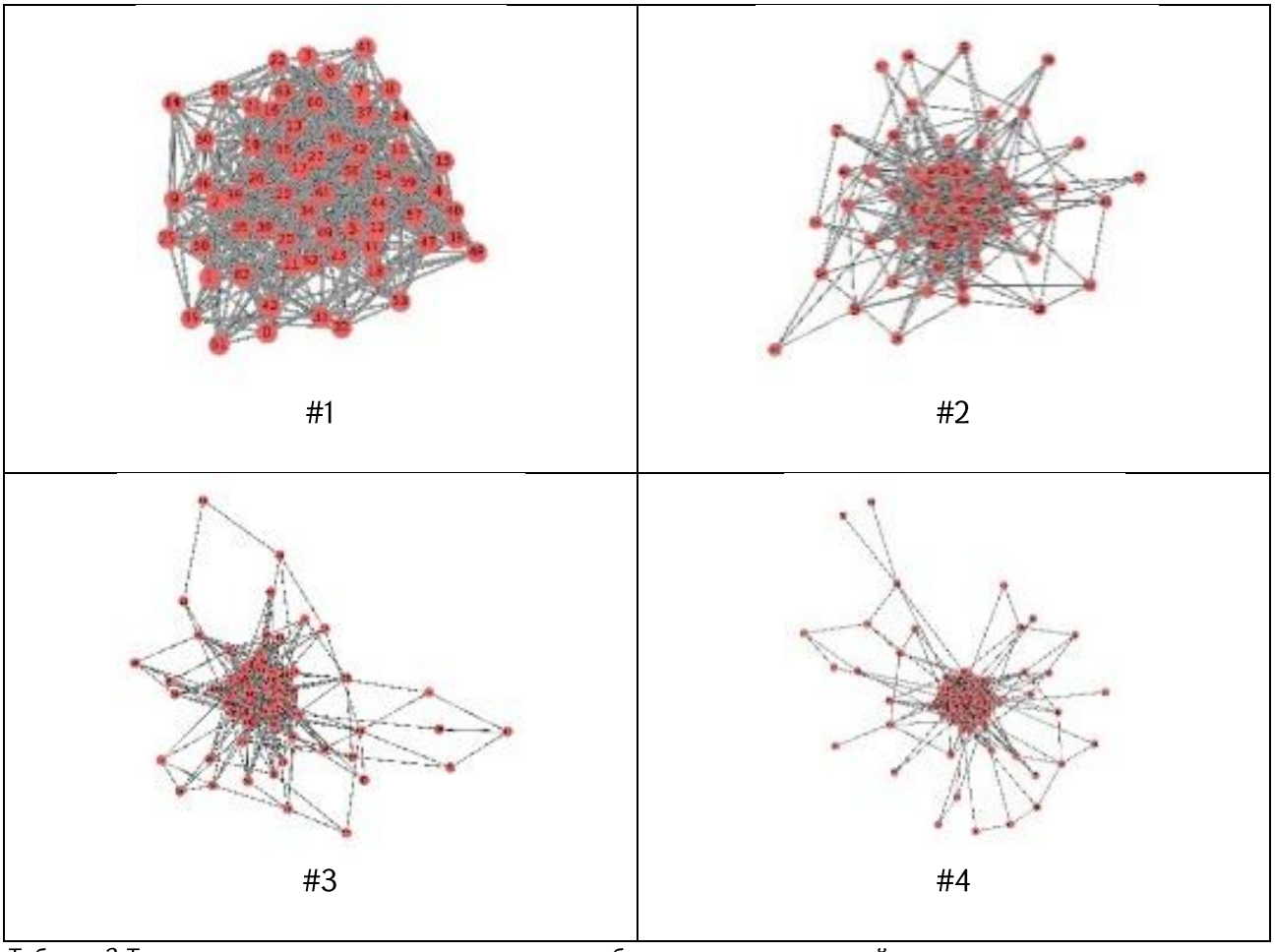


Таблица 3.Типичная картина развития сети в случае без сохранения степеней.

4.2.2. Структурообразование в сетях

Для получения структурной картины сетей была проведена их визуализация с помощью силового алгоритма Фрухтермана-Рейнгольда. Визуализация начальных сетей показала наличие

одного кластера, что и предполагалось для сетей модели Эрдёша-Реньи. Далее, производились визуализации сетей по ходу траектории. Далее приведены рисунки, показывающие структуру финальных сетей и картинки отделения кластеров на траектории для сети размером 256.

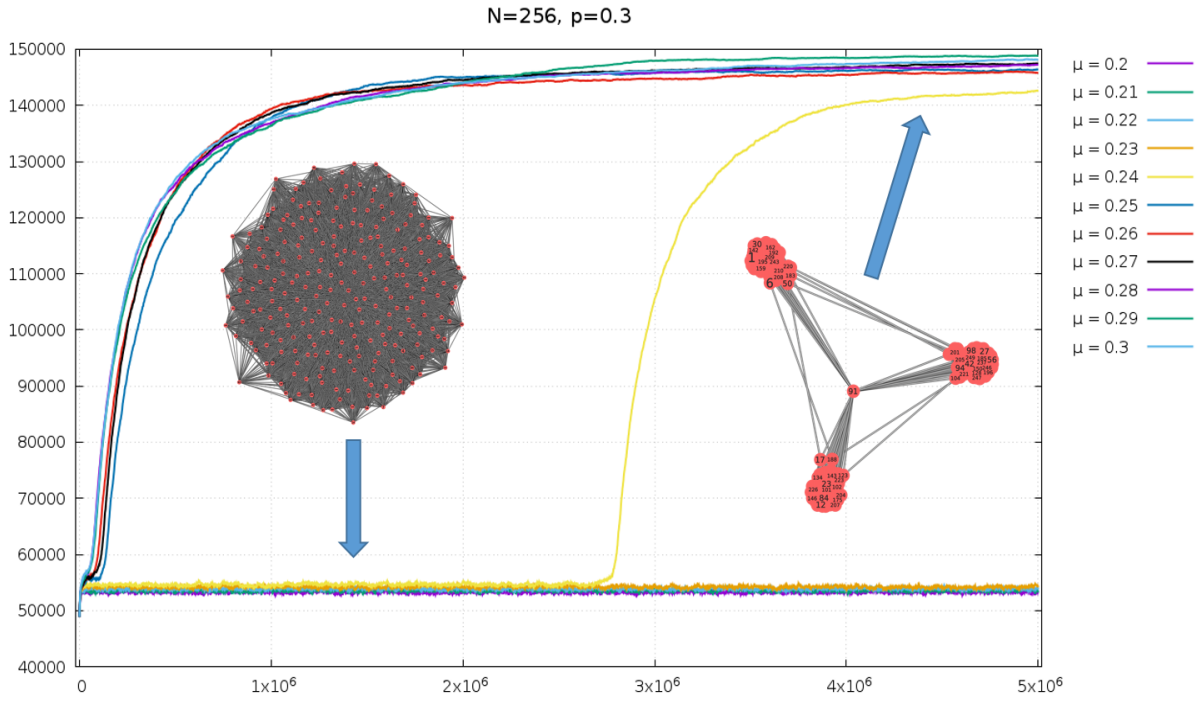


Рисунок 13. Структурная картина для случая N=256, p=0.3

Эволюция с сохранением степеней вершин приводит к реструктуризации сети и появлению кластерной структуры, причем количество кластеров обратно пропорционально вероятности связи p. Данный процесс можно рассматривать как образование многомасштабной архитектуры (в данном случае - двухуровневой). На первом уровне находятся плотные кластеры со слабыми межкластерными связями; на втором уровне узлами являются кластеры, а связи уже представляют связи между этими вершинами. Данная архитектура является отображением оптимальной структуры, необходимой для достижения поставленной цели – максимального количества циклов длины три в сети при наличии «вмороженного» беспорядка.

Следовательно, при наложении специфичных ограничений, процесс направленного развития может строить из совершенно случайной сети структуру с заранее известной топологией. Поэтому, процесс направленного развития сетей можно назвать целевым проектированием.

Каждый подъём сопровождается отщеплением нового почти полного кластера от общей массы вершин, не входящих в уже отделившиеся кластеры.

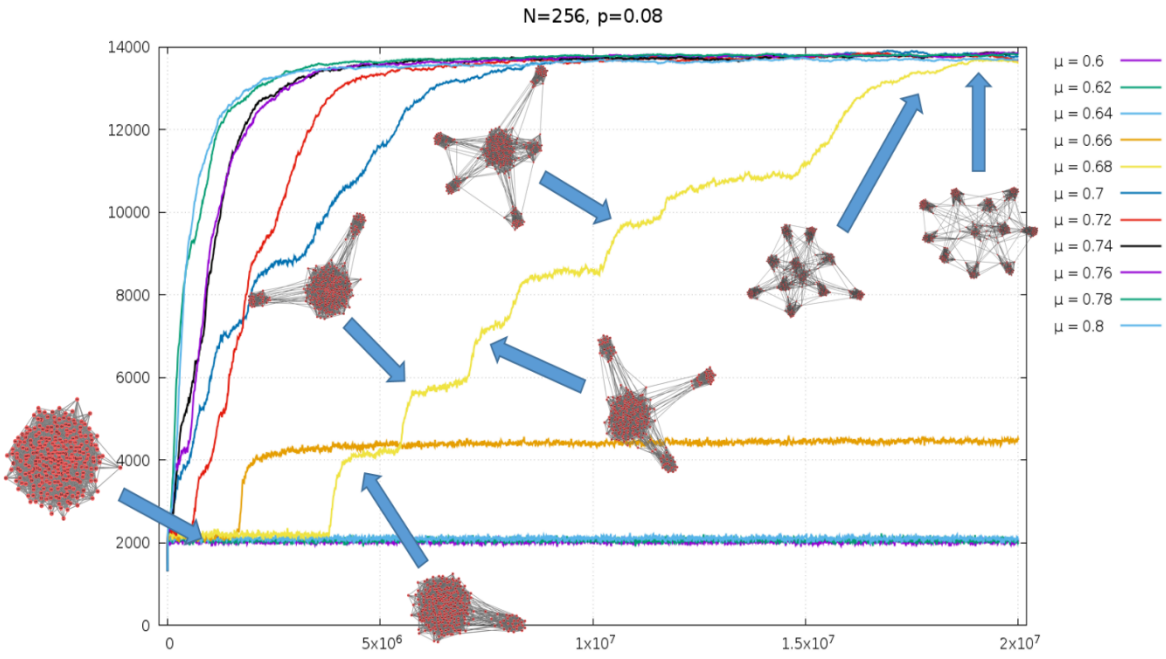


Рисунок 14. Структурная картина для случая N=256, p=0.08

4.2.3. Структура матриц смежности

Для более наглядного представления происходящих процессов были рассмотрены матрицы смежности сетей и динамика их изменения в ходе процесса. Была выполнена визуализация матриц смежности в виде квадратной сетки с закрашенными клетками, соответствующими связям в сети, т.е., если рассматривать численные значения матрицы, клетки со значением 0 оставлены пустыми, а клетки со значением 1 – закрашены в некоторый цвет. Матрицы получены по ходу процесса с определенным интервалом. Для получения наглядной картины, вершины сетей были перенумерованы следующим образом:

1) Рассматривается структура финальной сети и выделяются составляющие сеть кластеры. Для выявления кластеров используется алгоритм Louvain.
2) Производится перенумерация вершин финальной сети в соответствии с кластерной структурой сети, а именно, вершинам первого кластера ставятся в соответствие номера $1,...,n_1$, где n_1 – число вершин в первом кластере; вершинам второго кластера ставятся в соответствие номера $n_1+1,...,n_1+n_2$, где n_2 – число вершин во втором кластере; да-

лее, тот же процесс проводится для остальных кластеров. В результате получается отображение номеров вершин финальной сети с учетом кластерной структуры сети.

3) Полученное отображение применяется на все полученные матрицы.

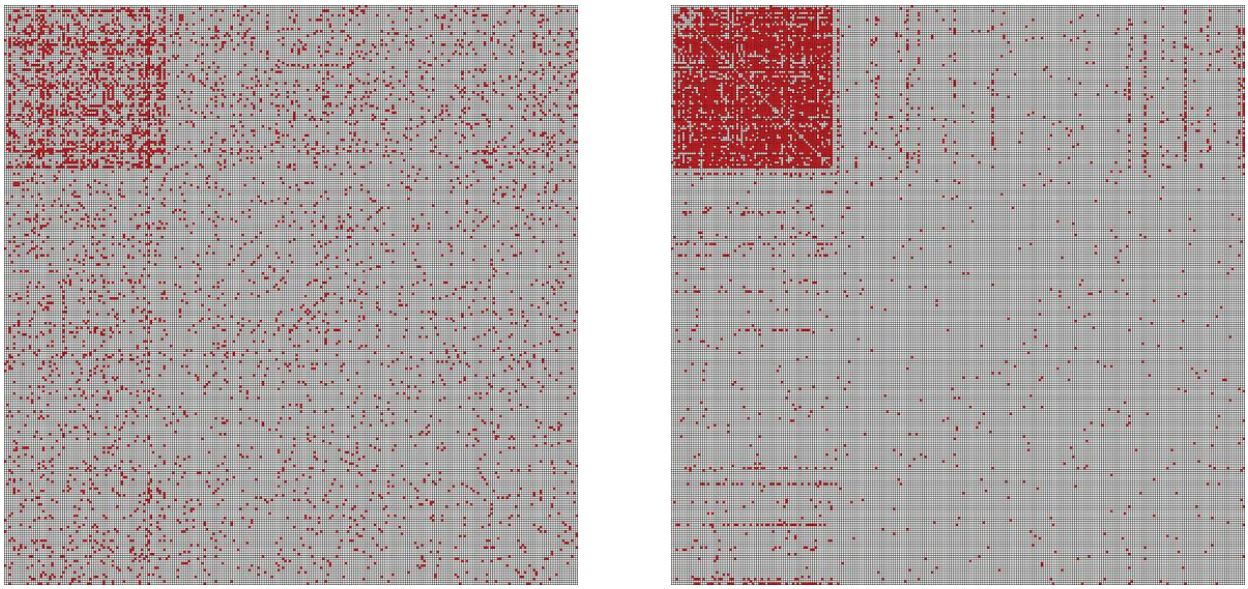


Рисунок 15. Структура матрицы смежности в ходе процесса эволюции без сохранения степеней вершин

Применение данного отображения позволяет графически представить структуру сети. Картина финальной сети показывает образовавшиеся в сети кластеры, а предыдущие – динамику их образования.

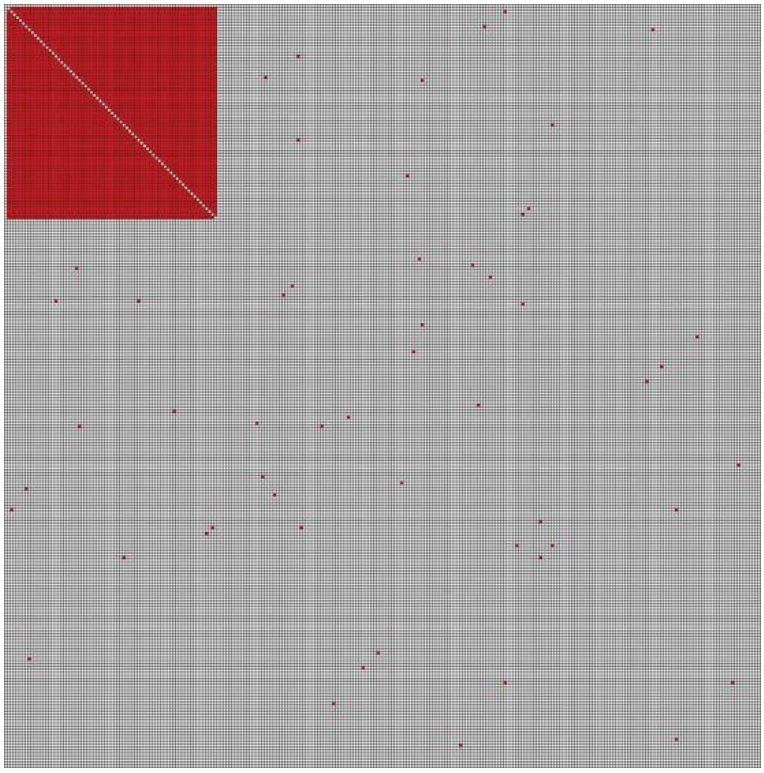


Рисунок 16. Финальная структура матрицы смежности в ходе процесса эволюции без сохранения степеней вершин

Данный процесс был проведен для матриц смежности сетей размером 256 для двух упомя-

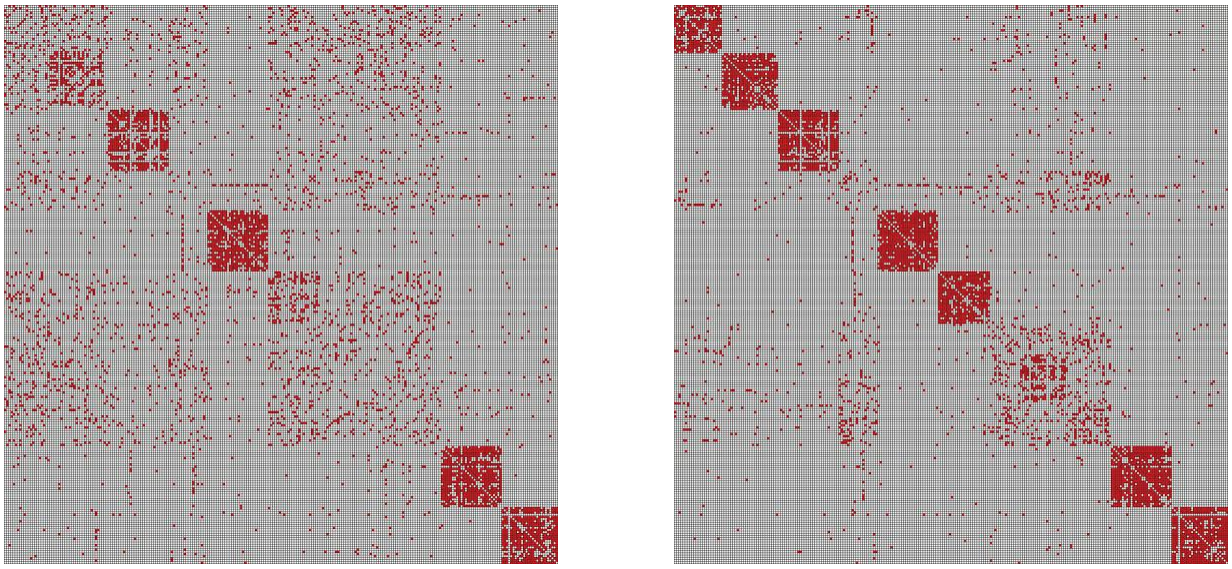


Рисунок 17. Структура матрицы смежности в ходе процесса эволюции с сохранением степеней вершин

нутых случаев процесса направленного развития ([1]).

В случае эволюции без сохранения степеней вершин наблюдается образование одного большого блока на диагонали. Данный блок является почти полным, что отображает большую плотность связей в образовавшемся кластере. Также видно небольшое количество разбросанных закрашенных клеток, что отображает наличие связей вне образовавшегося кластера – на-

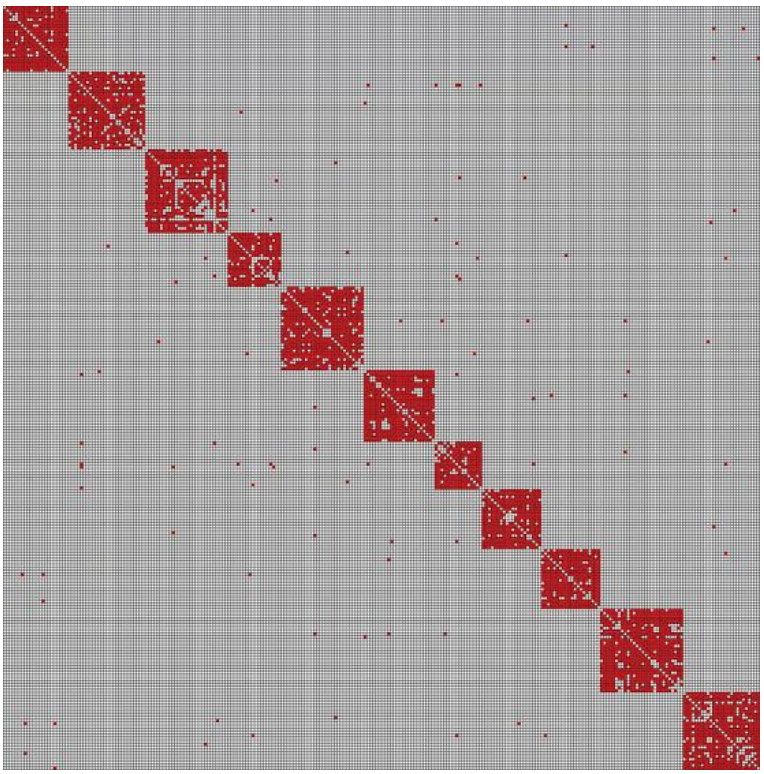


Рисунок 18. Финальная структура матрицы смежности в ходе процесса эволюции с сохранением степеней вершин.

званного «паром». Отсутствие закрашенных клеток в остальной части означает, что соответствующие вершины стали изолированными в результате процесса. Соответствующая картина приведена на рис. 15 и 16.

Случай с сохранением степеней обладает более богатой картиной. На диагонали образуются примерно равные по размерам блоки, каждый из которых представляет отдельный кластер. Блоки, как и в первом случае, почти полностью закрашены, что свидетельствует о большой плотности связей в блоках. Вне блоков очень мало закрашенных клеток и это свидетельствует о том, что связи между блоками слабые. Соответствующая картина приведена на рис. 17 и 18.

4.2.4. Спектр и спектральная плотность

Для дальнейшего исследования матриц смежности сетей, были рассмотрены спектр и спектральная плотность.

Спектральная плотность начальных сетей показала типичную для графов Эрдёша-Реньи картину – наблюдается две зоны, первая – основная – подчиняется закону полуокружностей

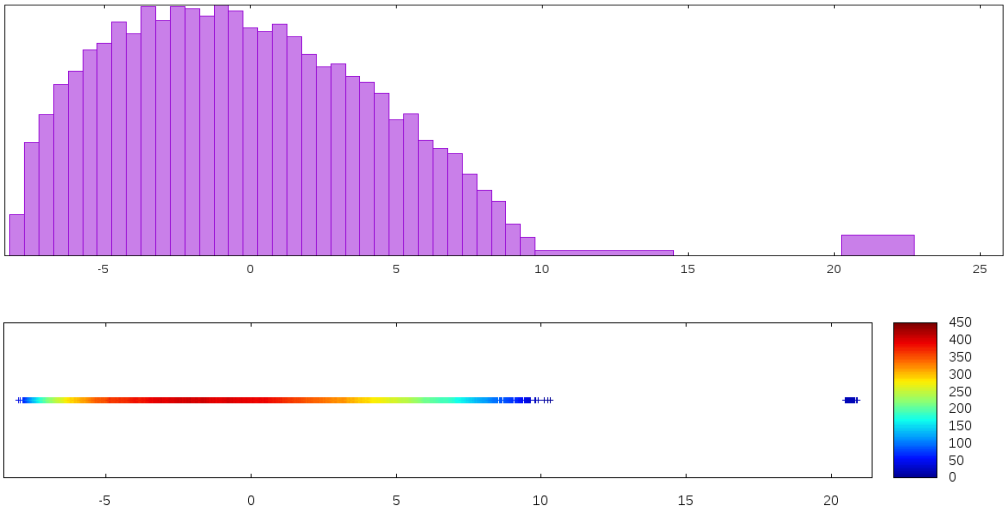


Рисунок 19. Спектральная плотность начальных сетей

Вигнера, а вторая – получается за счет отделенного от основной массы собственного значения, сигнализирующего о наличии единственного кластера в сети (рис. 19). Для визуализации использовались стандартные средства – гистограмма и тепловая карта спектральной плотности. Далее исследовались спектры финальных сетей и плотность объединённых спектров.

В случае процесса без сохранения степеней вершин при значениях параметра выше критического порога основная зона спектра сжимается и концентрируется вокруг отдельных значений. Спектральная плотность показывает, что в упомянутых точках спектра образуются пики. Отстоящее от общей массы собственных значений в начале процесса собственное значение в

конце процесса удаляется ещё дальше. Соответствующая картина приведена на рис. 20. На рисунке ясно видно, что пики на спектральной плотности распределены симметрично относительно некоторой точки и по мере удаления от этой точки плотность значений вокруг этих точек падает. Так как в процессе новых кластеров в сети не образуется, а существующий кластер только уменьшается и становится плотнее, то расстояние отдалённого собственного значения от основной массы говорит именно о плотности кластера.

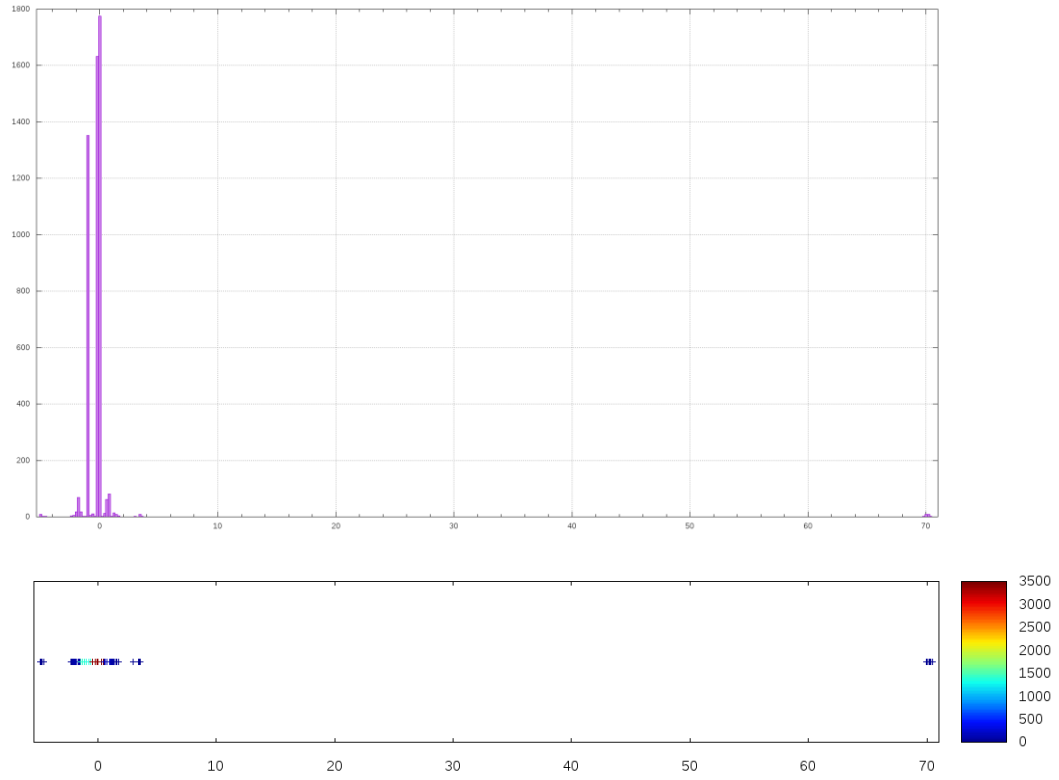


Рисунок 20. Финальная картина спектральной плотности в случае без сохранения степеней вершин

Для получения полной картины изменения спектра собственных значений в случае с сохранением степеней вершин, соответствующие значения были получены в окрестностях тех точек траектории, где наблюдается отщепление нового кластера. Визуализация показала, что образованию каждого кластера сопутствует отделение собственного значения от общей массы. Как следствие, в конце процесса, от основной массы значений отделяются k собственных значений, где k – число кластеров в финальной сети. Соответствующая картина приведена на рис. 21.

Спектральная плотность была рассмотрена на ансамблях из 20 случайных сетей Эрдеша-Реньи с одинаковыми параметрами генерации. На всех сетях был проведен процесс направленного развития с сохранением степеней вершин и получены финальные сети. Для каждой из

финальных сетей был получен спектр и, далее, все спектры были объединены в один объединенный спектр.

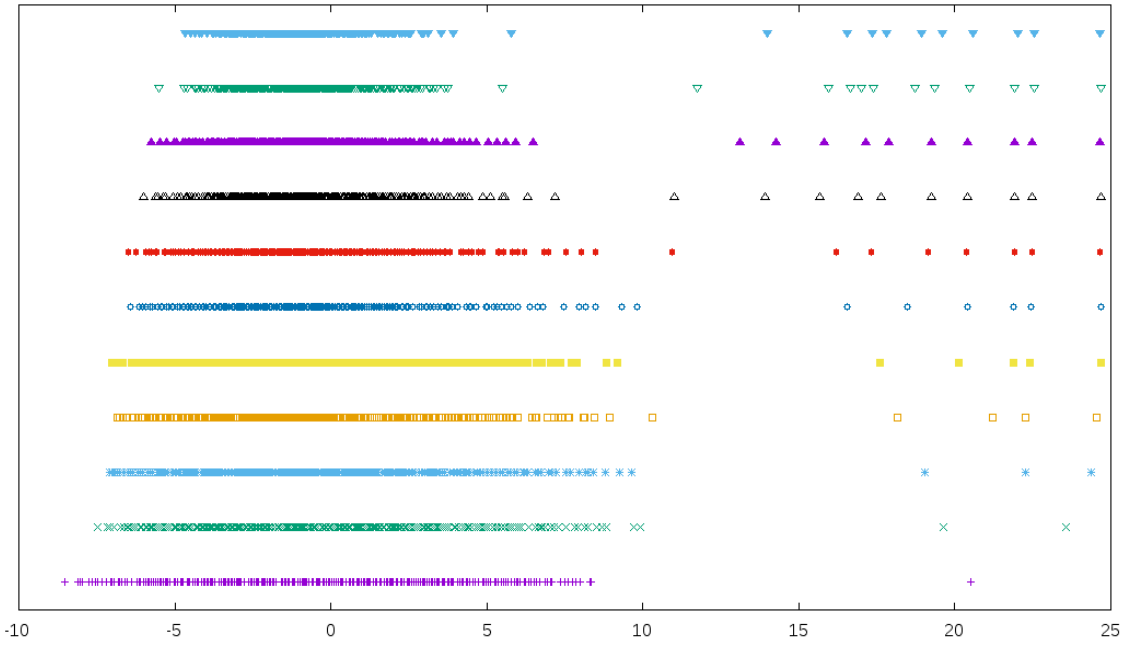


Рисунок 21. Динамика изменения спектра матриц смежности сети

Визуализация полученных для различных значений параметра μ объединенных спектров показала следующее:

- основная зона финальной спектральной плотности обладает треугольной формой (рис. 22), специфичной для масштабно-инвариантных сетей ([34])

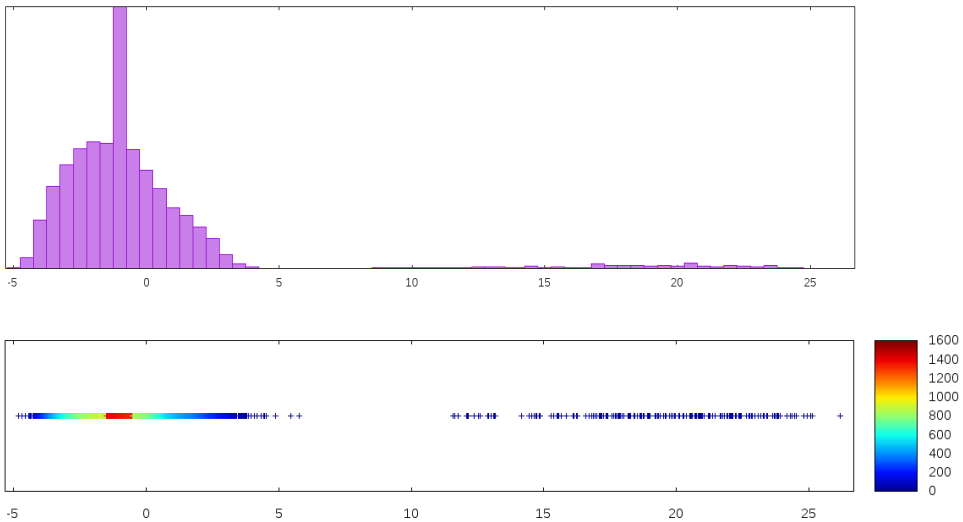


Рисунок 22. Финальная картина спектральной плотности в случае сохранения степеней вершин

- вторая зона – зона отделенных собственных значений, преобразуется из скопления нескольких значений вокруг одной точки в обширное плато, которое по размерам близко к основной зоне, однако обладает значительно меньшей плотностью (что демонстрируется гистограммой на рис. 22). Данная зона отображает кластерную структуру финальных сетей – как было сказано выше, оторванные собственные значения образовываются

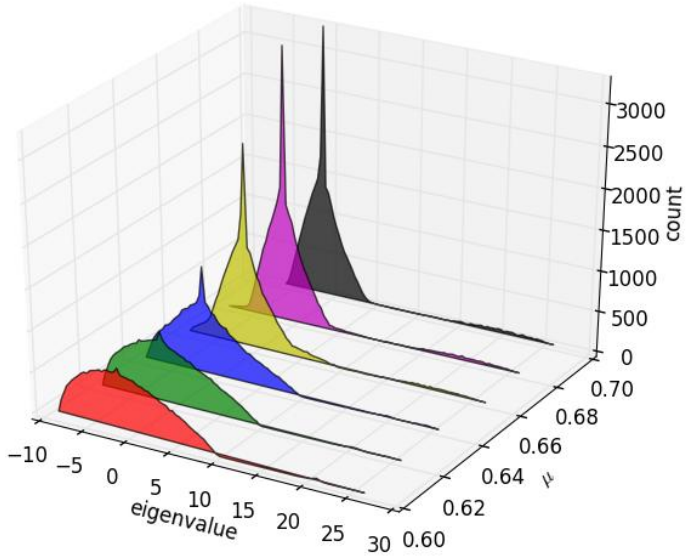


Рисунок 23. Динамика изменения спектральной плотности

параллельно с отщеплением кластеров от основной массы вершин в сети.

Для полноты картины, на рис. 23 и 24 приведена динамика изменения спектральной плот-

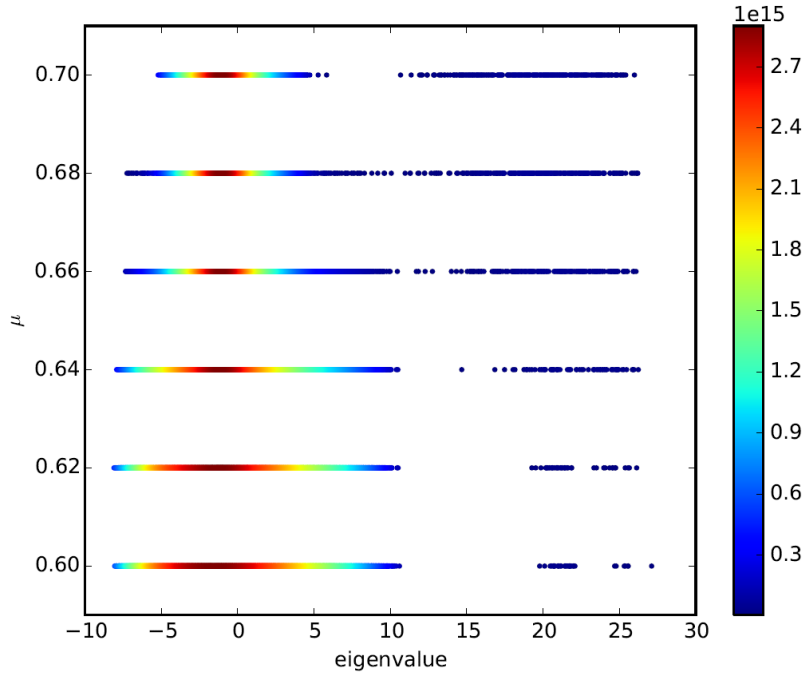


Рисунок 24. Динамика изменения спектральной плотности

ности при значениях вблизи критического порога в виде гистограммы и в виде тепловой карты, соответственно.

Как видно из рис. 23, спектральная плотность вблизи критического порога постепенно вытягивается, что приводит к сжатию и изменению формы основной зоны.

Для более глубокого понимания природы наблюдаемого изменения формы спектральной плотности были рассмотрены спектральные плотности каждого из образовавшихся в сети кластеров. Эксперименты показали, что спектральная плотность матриц смежности каждого из кластеров подчиняется закону полуокружностей Вигнера. Следовательно, изменение формы спектральной плотности обеспечивается межкластерными связями.

ЗАКЛЮЧЕНИЕ

Основные результаты

1. Разработаны алгоритмы и программные методы моделирования направленных процессов в сетях ([3]).

   Разработанные алгоритмы учитывают различные специфики процесса направленного развития и его рассматриваемых вариантов для обеспечения высокой производительности. Разработанные программные методы моделирования имеют своей целью обеспечение гибкости и возможности легкого расширения программных реализаций.

2. Разработана программная система, являющаяся универсальным инструментом для проведения экспериментальных исследований и разностороннего анализа процесса направленного развития сетей ([2] [3]).

   Система позволяет моделировать процесс направленного развития и получать динамику изменения рассматриваемых характеристик сетей. Для обеспечения высокой производительности в системе применяются параллельные и распределенные технологии и поддерживаются кластерные вычисления. Система спроектирована с использованием распространенных и подтвердивших свою эффективность программных техник, обеспечивающих её гибкость и расширяемость. Разработанная система позволила провести множество экспериментов по направленному развитию сетей, которые выявили представляющие интерес явления.

3. Исследован процесс направленного развития сетей с сохранением степеней узлов и без сохранения степеней узлов; получены основные характеристики сетей и их динамики в ходе двух рассматриваемых вариантов процесса развития сетей и проведен их сравнительный анализ. Получена структурная картина, образовывающаяся в сетях при направленном развитии ([1] [2]).

   С помощью разработанной системы проведены эксперименты для исследования процесса направленного развития сетей. Обработка и анализ начальных результатов позволили получить общую картину наблюдаемых в сетях явлений. На основе включенных в систему средств получения различных характеристик сетей получены основные характеристики финальных сетей и проведен сравнительный анализ двух вариантов переключения связей на каждом шаге процесса развития.

4. Исследованы матрицы смежности рассматриваемых сетей, получена общая картина наблюдаемой динамики. Исследованы спектры и спектральные плотности матриц смежности сетей ([1]).

   При помощи разработанных средств, позволяющих автоматизировать процесс визуализации сетей на различных шагах рассматриваемого процесса, была получена картина структурообразования в ходе процесса направленного развития сетей. На основе специальной техники визуализации матриц смежности сетей, реализованной в системе, удалось получить графическую картину динамики их структуры, что выявило множество деталей наблюдаемых явлений. С помощью автоматизированного процесса получения спектров и спектральных плотностей матриц смежности сетей была получена динамика их изменения и выявлены сходства свойств финальных сетей со свойствами масштабно-инвариантных сетей.

Возможные приложения результатов

Возможные приложения полученных результатов охватывают несколько областей. Далее представлены некоторые из них, указанные ведущими учеными соответствующих областей из научных учреждений России ([1]).

1) В последнее время матричную модель для больших N принято интерпретировать как теорию тахионов открытых струн на N нестабильных D0- или ZZ-бранах [89]. Финальная стадия эволюции нестабильной системы — это когерентное состояние режимов замкнутых струн или стабильных D-бран. В рассматриваемой модели наблюдается аналог данного явления – образование сильно когерентных состояний – набора кликов. Их число зафиксировано моделью с самого начала. Система на FZZT-бранах, известная как матричная модель Концевича [90], представляется актуальной для описания мультикликовой фазы рассматриваемой модели.
2) В контексте квантовой гравитации ([91] [92]) возникает следующий вопрос: рассматриваемая модель является топологической и не имеет метрической структуры – возможно ли трактовать кластеризацию как появление эффективной метрики ([93])? Можно предположить, что наблюдаемый фазовый переход соответствует переходу от топологической ($\langle g_{\mu\nu} \rangle = 0$) к геометрической фазе ($\langle g_{\mu\nu} \rangle \neq 0$) сети. Геометрическая фаза может иметь отношение к квантовой гравитации.

3) Другое применение связано с явлением почкования (образование похожих на пузырьки везикул из-за спонтанного искривления) в мембранах липидов [94]. Если мембрана жидкая, то материал может быть перераспределен по всей ткани и характерно образование только одной везикулы. Однако, в присутствии вмороженного беспорядка в мембране, перераспределение материала по всей ткани заблокировано и естественным видится образование фазы с присутствием множества везикул, аналогичное финальному состоянию сети в рассматриваемом процессе.

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

[1] V. Avetisov, M. Hovhannisyan, A. Gorsky, S. Nechaev, M. Tamm and O. Valba, "Eigenvalue tunneling and decay of quenched random network," *Physical Review E,* vol. 94, no. 6, pp. 062313 1-6, 2016.

[2] M. Hovhannisyan and S. Avetisyan, "Clustering of random networks under topological constraints," *Computer Science and Information Technologies (CSIT),* pp. 156-159, 2015.

[3] M. Hovhannisyan, "Parallel technologies in some problems of network modelling," *Proceedings of the YSU, Physical and Mathematical Sciences,* vol. 51, no. 1, pp. 53-59, 2017.

[4] A.-L. Barabási and M. Pósfai, Network Science, 1st ed., Cambridge University Press, 2016, p. 475.

[5] P. Erdős and A. Rényi, "On random graphs," *Publicationes Mathematicae,* vol. 6, pp. 290-297, 1959.

[6] M. Granovetter, "The strength of weak ties," *American Journal of Sociology,* vol. 78, pp. 1360-1380, 1973.

[7] D. J. Watts, Six Degrees: The Science of a Connected Age, W. W. Norton & Company, 2004, p. 384.

[8] S. Milgram, "The Small World Problem," *Psychology Today,* vol. 1, no. 1, pp. 61-67, May 1967.

[9] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature,* vol. 393, pp. 440-442, 1998.

[10] A. Clauset, C. R. Shalizi and M. E. J. Newman, "Power-Law Distributions in Empirical Data," *SIAM Rev.,* vol. 51, no. 4, pp. 661-703, 06 November 2009.

[11] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science,* vol. 286, pp. 509-512, 15 May 1999.

[12] L. Wu, B. N. Waber, S. Aral, E. Brynjolfsson and A. Pentland, "Mining face-to-face interaction networks using sociometric badges: Predicting productivity in an it configuration task," in *Proceedings of the International Conference on Information Systems*, Paris, 2008.

[13] A.-L. Barabási and Z. N. Oltvai, "Network Biology: Understanding the cell's functional organization," *Nature Reviews Genetics,* vol. 5, no. 2, pp. 101-113, March 2004.

[14] A.-L. Barabási, N. Gulbahce and J. Loscalzo, "Network medicine: a network-based approach to

human disease," *Nature Reviews Genetics,* vol. 12, pp. 56-68, January 2011.

[15] J. Arquilla and D. Ronfeldt, Networks and Netwars: The Future of Terror, Crime, and Militancy, RAND Corporation, 2001, p. 380.

[16] L. Hufnagel, D. Brockmann and T. Geisel, "Forecast and control of epidemics in a globalized world," *PNAS,* vol. 101, no. 42, pp. 15124-15129, 19 October 2004.

[17] P. Wang, M. C. González, C. A. Hidalgo and A.-L. Barabási, "Understanding the Spreading Patterns of Mobile Phone Viruses," *Science,* pp. 1071-1076, 22 May 2009.

[18] O. Sporns, G. Tononi and R. Kötter, "The Human Connectome: A Structural Description of the Human Brain," *PLoS Computational Biology,* vol. 1, no. 4, pp. 0245-0251, September 2005.

[19] E. N. Gilbert, "Random Graphs," *Annals of Mathematical Statistics,* vol. 30, no. 4, pp. 1141-1144, 1959.

[20] P. Erdős and A. Rényi, "On the Evolution of Random Graphs," *Publications of the Mathematical Institute of the Hungarian Academy of Sciences,* vol. 5, pp. 17-61, 1960.

[21] B. Bollobás, "6. The Evolution of Random Graphs—The Giant Component," in *Random Graphs (Cambridge Studies in Advanced Mathematics)*, 2nd ed., Cambridge University Press, 2001, pp. 130-159.

[22] M. Molloy and B. Reed, "A critical point for random graphs with a given degree sequence," *Random Structures & Algorithms,* vol. 6, no. 2-3, pp. 161-180, March-May 1995.

[23] I. Kryven, "Emergence of the giant weak component in directed random graphs with arbitrary degree distributions," *Physical Review E,* vol. 94, no. 1, p. 012315, July 2016.

[24] M. E. J. Newman, S. H. Strogatz and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review E,* vol. 64, no. 2, pp. 261181-261187, August 2001.

[25] D. Stauffer and A. Aharony, Introduction to Percolation Theory, 2nd ed., CRC Press, 1994, p. 192.

[26] S. R. Broadbent and J. M. Hammersley, "Percolation processes," *Mathematical Proceedings of the Cambridge Philosophical Society,* vol. 53, no. 03, p. 629, 2008.

[27] G. R. Grimmet and J. M. Marstrand, "The Supercritical Phase of Percolation is Well Behaved," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences,* vol. 430, no. 1879, pp. 439-457, 1990.

[28] A.-L. Barabási and R. Albert, "Statistical mechanics of complex networks," *Reviews of Modern Physics,* vol. 74, no. 1, pp. 47-97, 2002.

[29] A. Barrat and M. Weigt, "On the properties of small-world network models," *European Physical Journal B,* vol. 13, no. 3, pp. 547-560, 2000.

[30] M. Steyvers and J. B. Tenenbau,, "The Large-Scale Structure of Semantic Networks: Statistical Analyses and a Model of Semantic Growth," *Cognitive Science,* vol. 29, no. 1, pp. 41-78, 2005.

[31] K. Soramäki, "The topology of interbank payment flows," *Physica A: Statistical Mechanics and its Applications,* vol. 379, no. 1, pp. 317-333, 2007.

[32] M. Fratini, N. Poccia, A. Ricci, G. Campi, M. Burghammer, G. Aeppli and A. Bianconi, "Scale-free structural organization of oxygen interstitials in La2CuO4+y," *Nature,* vol. 466, no. 7308, pp. 841-844, 2010.

[33] R. Cohen and S. Havlin, "Scale-Free Networks Are Ultrasmall," *Physical Review Letters,* vol. 90, no. 5, p. 058701, 2003.

[34] K.-I. Goh, B. Kahng and D. Kim, "Spectra and eigenvectors of scale-free networks," *Physical Review E,* vol. 64, p. 051903, 2001.

[35] U. Alon, "Network motifs: theory and experimental approaches," *Bature Reviews Genetics,* vol. 8, pp. 450-451, June 2007.

[36] R. Milo, S. S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii and U. Alon, "Network motifs: simple building blocks of complex networks," *Science,* vol. 298, no. 5594, pp. 824-827, 2002.

[37] S.-O. S. Shai, R. Milo, S. Mangan and U. Alon, "Network motifs in the transcriptional regulation network of Escherichia coli," *Nature Genetics,* vol. 31, pp. 64-68, 22 April 2002.

[38] A. Masoudi-Nejad, F. Schreiber and Z. R. M. Kashani, "Building blocks of biological networks: a review on major network motif discovery algorithms," *IET Systems Biology,* vol. 6, no. 5, pp. 164-174, 16 November 2012.

[39] S. Mangan and U. Alon, "Structure and function of the feed-forward loop network motif," *PNAS,* vol. 100, no. 21, p. 11980–11985, 14 October 2003.

[40] S. Mangan, A. Zaslaver and U. Alon, "The Coherent Feedforward Loop Serves as a Sign-sensitive Delay Element in Transcription Networks," *Journal of Molecular Biology,* vol. 2334, no. 2, pp. 197-204, 21 November 2003.

[41] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer and U. Alon,

"Superfamilies of designed and evolved networks," *Science,* vol. 303, no. 5663, pp. 1538-1542, 2004.

[42] S. N. Dorogovtsev and J. F. F. Mendes, "Evolution of networks," *Advances in Physics,* vol. 51, no. 4, pp. 1079-1187, 2002.

[43] M. V. Tamm, A. B. Shkarin, V. A. Avetisov, O. V. Valba and S. K. Nechaev, "Islands of Stability in Motif Distributions of Random Networks," *Physical Review Letters,* vol. 113, no. 9, p. 09501, 26 August 2014.

[44] V. A. Avetisov, S. K. Nechaev and A. B. Shkarin, "On the motifs distribution in random hierarchical networks," *Physica A: Statistical Mechanics and its Applications,* vol. 389, no. 24, pp. 5895-5902, 15 December 2010.

[45] J. Gao, B. Barzel and A.-L. Barabási, "Universal resilience patterns in complex networks," *Nature,* vol. 530, pp. 307-312, 18 February 2016.

[46] S. H. Strogatz, "From Kuramoto to Crawford: Exploring the onset of synchronization in populations of coupled oscillators," *Physica D,* vol. 143, pp. 1-20, 2000.

[47] A. Arenas, A. Díaz-Guilera, J. Kurths, Y. Moreno and C. Zhou, "Synchronization in complex networks," *Physics Reports,* vol. 469, no. 2, pp. 93-153, December 2008.

[48] A. L. Lloyd and R. M. May, "How viruses spread among computers and people," *Science,* vol. 292, no. 5520, pp. 1316-1317, 18 May 2001.

[49] M. A. M. de Aguiar and Y. Bar-Yam, "Spectral analysis and the dynamic response of complex networks," *Physical Review E,* vol. 71, no. 1, p. 016106, 4 January 2005.

[50] L. O. Bartoneva, S. A. Basharin, M. Y. Bychkov, A. A. Kal'nin and A. V. Korolev, "Collective effects in networks of negatron elements," *Technical Physics,* vol. 43, no. 5, pp. 477-483, May 1998.

[51] S. N. Dorogovtsev, A. V. Goltsev and J. . F. F. Mendes, "Critical phenomena in complex networks," *Reviews of Modern Physics,* vol. 80, no. 4, pp. 1275-1335, 6 October 2008.

[52] M. Bernaschi, F. Castiglione, A. Ferranti, C. Gavrila, M. Tinti and G. Cesareni, "ProtNet: a tool for stochastic simulations of protein interaction networks dynamics," *BMC Bioinformatics,* vol. 8, pp. 1-11, 2007.

[53] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson and C. Guestrin, "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs," in *10th USENIX Symposium on Operating Systems*

Design and Implementation, Hollywood, CA, 2012.

[54] «Случайный граф,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/Случайный_граф.

[55] B. Bollobás and O. M. Riordan, "Chapter 1. Mathematical results on scale-free random graphs," in *Handbook of Graphs and Networks: From the Genome to the Internet*, Weinheim, Wiley-VCH Verlag GmbH & Co. KGaA, 2002, pp. 1-34.

[56] «Случайные графы,» [В Интернете]. Available: http://rain.ifmo.ru/cat/view.php/theory/graph-general/random-2005.

[57] B. Bollobás, Random Graphs, 2nd ed., Cambridge: Cambridge University Press, 2001, p. 518.

[58] O. Riordan and N. Wormald, "The Diameter of Sparse Random Graphs," *Combinatorics, Probability and Computing,* vol. 19, no. 5-6, pp. 835-926, 1 November 2010.

[59] F. Harary, Graph Theory, Addison-Wesley Pub. Co., 1969, p. 274.

[60] З. В. Апанович, «Современные силовые алгоритмы для визуализации информации большого объёма,» в *XIV Международная конференция «Проблемы управления и моделирования в сложных системах» (ПУМСС-2012)*, Самара, 2012.

[61] P. Eades, "A heuristic for graph drawing," *Congressus Nutnerantiunt,* vol. 42, pp. 149-160, 1984.

[62] P. Eades, "Fixed edge-length graph drawing is NP-hard," *Discrete Applied Mathematics,* vol. 28, no. 2, pp. 111-134, August 1990.

[63] T. M. J. Fruchterman and E. M. Reingold, "Graph Drawing by Force-Directed Placement," *Software - Practice and Experience,* vol. 21, no. 11, pp. 1129-1164, 1991.

[64] З. В. Апанович, «Методы визуализации информации при помощи графов. Часть 2. Методы визуализации ориентированных и неориентированных графов,» [В Интернете]. Available: http://mmc2.nsu.ru/default.aspx?db=book_apanovich2&int=VIEW&el=1684&templ=l206.

[65] M. A. Porter, J.-P. Onnela and P. J. Mucha, "Communities in Networks," *Notices of the American Mathematical Society,* vol. 56, no. 9, pp. 1082-1097, 1164-1166, 2009.

[66] V. D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment,* vol. 10, 9 October 2008.

[67] «Инвариант графа,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/Инвариант_графа.

[68] E. P. Wigner, "Characteristic Vectors of Bordered Matrices With Infinite Dimensions," *Annals of Mathematics,* vol. 62, no. 3, pp. 548-564, November 1955.

[69] E. P. Wigner, "On the Distribution of the Roots of Certain Symmetric Matrices," *Annals of Mathematics,* vol. 67, no. 2, pp. 325-327, March 1958.

[70] «Алгоритм Метрополиса — Гастингса,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/Алгоритм_Метрополиса_—_Гастингса.

[71] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics,* vol. 21, no. 6, pp. 1087-1092, June 1953.

[72] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika,* vol. 57, no. 1, pp. 97-109, 1 April 1970.

[73] F. Viger and M. Latapy, "Efficient and simple generation of random simple connected graphs with prescribed degree sequence," *Lecture Notes in Computer Science,* vol. 3595, pp. 440-449, 2005.

[74] S. Maslov and K. Sneppen, "Specificity and stability in topology of protein networks," *Science,* pp. 910-913, 3 May 2002.

[75] E. Dekel and U. Alon, "Optimality and evolutionary tuning of the expression level of a protein," *Nature,* vol. 436, no. 7050, pp. 588-592, 2005.

[76] U. Alon, M. Evans, H. Hinrichsen and D. Mukamel, "Roughening transition in a one-dimensional growth process," *Physical Review Letters,* vol. 76, no. 15, p. 2746, 8 April 1996.

[77] O. Frank and D. Strauss, "Markov Graphs," *Journal of the American Statistical Association,* vol. 81, no. 395, pp. 832-842, September 1986.

[78] Z. Burda, J. Jurkiewicz and A. Krzywicki, "Network transitivity and matrix models," *Physical Review E,* vol. 69, no. 2, p. 026106, 20 February 2004.

[79] Z. Burda, J. Jurkiewicz and A. Krzywicki, "Perturbing general uncorrelated networks," *Physical Review E,* vol. 70, no. 2, p. 026106, 16 August 2004.

[80] К. Хьюз и Т. Хьюз, Параллельное и распределенное программирование с использованием C++, Издательский дом "Вильямс", 2004, p. 672.

[81] М. Левин, Параллельное программирование с использованием OpenMP, 2-е ред., Москва: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2012,

p. 121.

[82] «Nvidia CUDA — неграфические вычисления на графических процессорах,» [В Интернете]. Available: http://www.ixbt.com/video3/cuda-1.shtml.

[83] В. П. Гергель, Теория и практика параллельных вычислений, Москва: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007, p. 423.

[84] И. П. Корнфельд, Я. Г. Синай и С. В. Фомин, Эргодическая теория, Москва: Наука, 1980, p. 384.

[85] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, 1st ed., Addison-Wesley Professional, 1994, p. 395.

[86] G. Booch, Object-Oriented Analysis and Design with Applications, 3rd ed., Addison Wesley, 2004, p. 720.

[87] D. B. Giuseppe, P. Eades, R. Tamassia and I. G. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, 1st ed., Prentice Hall, 1998, p. 397.

[88] S. G. Kobourov, "Spring Embedders and Force-Directed Graph Drawing Algorithms," *https://arxiv.org/abs/1201.3011,* 2012.

[89] I. R. Klebanov, J. Maldacena and N. Seiberg, "D-brane decay in two-dimensional string theory," *Journal of High Energy Physics,* vol. 2003, no. JHEP07, 21 July 2003.

[90] D. Gaiotto and L. Rastelli, "A paradigm of open/closed duality Liouville D-branes and the Kontsevich model," *Journal of High Energy Physics,* vol. 2005, no. JHEP07, 22 July 2005.

[91] G. Bianconi and C. Rahmede, "Network geometry with flavor: From complexity to quantum geometry," *Physical Review E,* vol. 93, no. 3, p. 032315, 14 March 2016.

[92] C. A. Trugenberger, "Quantum gravity as an information network self-organization of a 4D universe," *Physical Review D,* vol. 92, no. 8, p. 084014, 5 October 2015.

[93] D. Krioukov, "Clustering Implies Geometry in Networks," *Physical Review Letters,* vol. 116, no. 20, p. 1208302, 19 May 2016.

[94] R. Lipowsky, "Budding of membranes induced by intramembrane domains," *Journal de Physique II,* vol. 2, no. 10, pp. 1825-1840, October 1992.

УТВЕРЖДАЮ

Заместитель директора
Федерального государственного
бюджетного учреждения науки
Института химической физики им.
Н.Н.Семенова Российской академии наук
(ИХФ РАН)
дфмн, профессор

Крупянский Ю.Ф.

"7" апреля 2017г.

В специализированный совет 037
"Информатика и вычислительные
системы"
Института проблем информатики и
автоматизации НАН РА

ул. П. Севака 1, 0014 г. Ереван,
Республика Армения

Акт

о внедрении результатов диссертационной работы
Оганесяна Минаса Генриковича
«Целевое проектирование сетевых систем многомасштабной
архитектуры с использованием распределенных вычислительных
ресурсов»

Настоящим подтверждается, что программная среда, спроектированная и разработанная в рамках диссертационной М. Г. Оганесяна «Целевое проектирование сетевых систем многомасштабной архитектуры с использованием распределенных вычислительных ресурсов», внедрена и используются в Институте химической физики им. Н.Н. Семенова Российской академии наук. Разработанная Оганесяном М. Г. система позволила провести новаторские исследования и получить результаты фундаментальной значимости, относящихся к стохастическим процессам формирования сложных сетей, выявления их общих закономерностей и возникающих структурных архетипов. Разработанная Оганесяном программная среда показала свою высокую эффективность и имеет широкий потенциал применения в междисциплинарных фундаментальных и прикладных исследования на границах физики, химии и биологии.

Руководитель работ,
зав. лабораторией
теории сложных систем
дфмн

Аветисов В. А.