Содержание

ВВЕДЕНИЕ

Исследователи, изучающие свойства живой материи, социальных систем, крупных техногенных объектов, все чаще приходят к выводу, что знания о свойствах отдельных элементов (гены, нейроны, банки, дорожные перекрестки) недостаточны для сколько-нибудь полного понимания функционирования соответствующих сложных систем в целом (живой клетки, мозга, экономики, транспорта). В самом деле, у больных раком нет отдельного «гена рака», а есть мутации в нескольких десятков взаимодействующих генов. Сознание не может быть сведено к поведению отдельного нейрона, а требует учета взаимодействия миллиардов синапсов. Экономический кризис не будет понят без изучения сложной паутины взаимных задолженностей, свойственных мировой финансовой системе. Изучение причин образования дорожных пробок и нахождение решений для их предотвращения требует изучения структуры всей дорожной сети города. В этих науках, наряду с парадигмой редукционизма, которая еще отнюдь не исчерпала свой потенциал, должна иметь место новая парадигма, отражающая сложность соответствующих систем. Вот почему часто вопросы из разных областей науки и жизни приводятся к изучению некоторого вида процесса какого-то сети, с помощью которого можно будет однозначно и очень глубоко описать нужный процесс или вопрос.

Один из распространенных областей использования теории сети является ‘deep learning’, на основе которого лежит многослойная нейронная сеть, вершины распространены по иерархии и в процессе активные вершины текущего уровня пытаются активизировать каждую вершину следующего уровня передавая ей сумму весов соответствующих ребер, а процесс определения весов и является главным этапом обучении системы [6]. Другой областью использования теории сети – астрономия: астрономам важно знать, как звезды в звездной системе распределяются по массам. Звезды взаимодействуют друг с другом внутри скоплений, и для того, чтобы понимать, как происходит это взаимодействие, нужны данные о том, сколько звезд и какой массы входят в систему. Вместо того чтобы описывать процесс роста каждой индивидуальной протозвезды, исследователи представили всю систему как пространственную сеть и использовали математические методы теории сетей. В результате удалось показать, что даже если изначально протозвёзды имели бы случайную массу, дальнейшее взаимодействие в рамках такой случайно образующейся сети неизбежно ведёт к "выравниванию" их распределения масс по степенному закону. С помощью всего восьми уравнений и без привлечения каких-либо дополнительных параметров исследователям довольно элегантно удалось теоретически объяснить начальную функцию масс звёзд [5].

Основной целью этой работы является исследование процесса распространения активности в сетях. В рамках этого документа все исследования сделаны для случайных сетей модели Эрдеша-Реньи.

Нейронная сеть состоит из вершин, называемым нейронами и ребер, связывающие пары вершин. Каждый нейрон может находиться в одном из двух состояний: активный и пассивный. Во время своей деятельности нейроны взаимодействуют друг с другом - в случае активности передают импульс, пытаясь активировать своих соседей, после чего сами могут становиться пассивным. Один из главных вопросов этого процесса является нахождение критического порога. Гипотеза критичности утверждает, что биологические системы могут выполнять сложные вычисления, которые необходимы, чтобы выжить только за счет работы при критичности, то есть на границе между активной или хаотической фазой, где шум распространяется безгранично, тем самым искажая всю обработку информации, или фазой в которой возмущения быстро исчезают, препятствуя способности реагировать и адаптироваться. Критическая динамика обеспечивает деликатный компромисс между этими двумя непрактичными тенденциями, и было высказано мнение, которое подразумевает оптимальную передачу и хранение информации, оптимальные вычислительные возможности, большую устойчивость сети, максимальное разнообразие видов памяти, а также максимальную чувствительность к раздражителям. Недавний анализ головного мозга человека в состоянии покоя, показывает, что мозг проводит большую часть времени блуждая вокруг широкой области вблизи критической точки, а не просто сидит на нем. Это говорит о том, что область, где корковые сети работают, является не просто критической точкой, а целая расширенная область вокруг него [2][3].

В первой главе данной работы рассматриваются теоретические основы исследований: описание случайной сети, основные концепции моделей случайной сети, модель Эрдеша – Реньи и его свойств. Во второй главе описываются процесс распространения активности в сети, три основных алгоритма для реализации этого процесса и подходы для тестирования каждого алгоритма. В третьей главе описываются этапы процесса исследования. В четвертой главе приведены результаты анализа процесса распространения активности в случайных сетях. В пятой главе представлены основные характеристики программной системы. В заключении обсуждаются возможные дальнейшие исследования. В первом приложении приведены графические представления результатов алгоритма ‘B’ для случаев, когда во время процесса включена только а) активация, б) деактивация. Во втором приложении приведена визуализация процесса распространения активности для сети с параметрами {N=65, p = 0.05}

Глава 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

1.1. Сети и графы

В математических терминах сеть представляется как граф. Граф $G = (V, E)$ совокупность двух множеств: множества N узлов (вершин или точек) $V = \{v_1, v_2, \dots, v_N\}$ и множеством ребер (соединений или линий) $E = \{e_1, e_2, \dots, e_M\}$, которые соединяют два элемента из V. Если ребра имеют направлении, то граф G является ориентированным - $\vec{G}(V, \vec{E})$. Если ребра, соединяющие вершины, не имеют направления, то граф является неориентированным. Если v_1, v_2 - вершины, а e=(v_1, v_2) - соединяющее их ребро, тогда вершина v_1, и ребро *e инцидентны*, вершина v_2 и ребро *e* тоже инцидентны.[4]

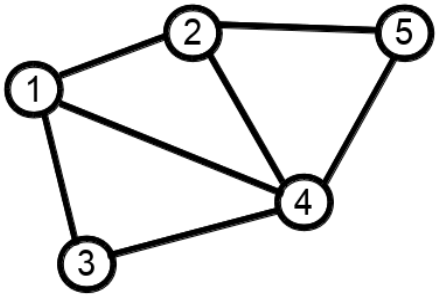


Рис. 1. Иллюстрация графа с $N = 5$ вершинами и $M = 7$ ребрами. Множество вершин: V= $\{1, 2, 3, 4, 5\}$. Множество ребер: $E = \{\{1, 2\}, \{1,3\}, \{1,4\}, \{2,5\}, \{4,5\}, \{2,4\}, 3,4\}\}$.

Для представления графа есть три основных способа:

1) *Матрица смежности* - один из способов представления графа в виде матрицы. Пусть имеется *G* граф с конечным числом вершин N (пронумерованных числами от 1 до N). Соответствующая матрица будет квадратная матрица размера N, в которой значение элемента a_{ij} равно числу ребер из i-ой вершины графа в j-ю вершину. В частности, матрица смежности простого графа (не содержащего петель и кратных рёбер) является бинарной матрицей и содержит нули на главной диагонали.

Таб. 1: Матрица смежности для графа из Рис. 1

0	1	1	1	0
1	0	0	1	1
1	0	0	1	0
1	1	1	0	1
0	1	0	1	0

2) Список смежности — один из способов представления графа в виде коллекции списков вершин. Каждой вершине графа соответствует список, состоящий из "соседей" этой вершины.

Таб. 2: Список смежности для графа из Рис. 1

{2, 3, 4}
{1,4,5}
{1,4}
{1,2,3,5}
{2,4}

3) *Список ребер* – в этом случае для каждого ребра графа хранится пара номеров вершин, которые оно соединяет. Но в этом случае граф должен быть связанным. Представление графа из рис.1 с помощью списку ребер будет:

{{1, 2}, {1,3}, {1,4}, {2,5}, {4,5}, {2,4}, 3,4}}.

Степенью или *валентностью* вершины неориентированного графа называется число ребер, инцидентных данной вершине.

Маршрут в графе — это чередующаяся последовательность вершин и рёбер, в которой любые два соседних элемента инцидентны. Если начальная вершина является и последним в маршруте - то маршрут *замкнут*. Граф называется *связным*, если любая пара его вершин соединена маршрутом.

Малые миры. Концепция малых миров довольно просто описывает тот факт, что, несмотря на их огромные размеры, в большинстве сетей существует сравнительно короткий путь между двумя любыми вершинами. Расстояние между двумя вершинами определяется как число ребер наикратчайшего пути, их соединяющего. Но, принцип «малых миров», хотя и интригует, не является специальным принципом организации. Наконец, Эрдёшем и Реньи показано, что среднее расстояние между двумя вершинами в случайном графе растет как логарифм от числа вершин и таким образом, случайные графы также являются маленькими мирами.

Кластеризация – это локальная характеристика сети. Она характеризует степень взаимо- действия между собой ближайших соседей данного узла. В большинстве сетей, если узел A соединен с узлом B, а узел B – с узлом C, то существует большая вероятность, что узел A соединен с узлом C (друзья наших друзей обычно также являются и нашими друзьями). Коэффициент кластеризации данного узла есть вероятность того, что два ближайших соседа этого узла сами есть ближайшие соседи[8]. Другими словами, если узел j

имеет q_j ближайших соседей с числом t_j связей между ними, то локальный коэффициент кластеризации равен

$$C_j(q_j) = \frac{t_j}{q_j(q_j-1)/2},$$

а среднее значение кластеризации по всем узлам - $(C) = \frac{1}{N}\sum_j C_j$. Таким образом, кластеризация характеризует статистику циклов (треугольников) в сети. Важно то, что большинство реальных сетей в мире обладают высокой кластеризацией, на порядки превышающей кластеризацию классических случайных сетей в модели Ердеша–Реньи

$$C(q) \cong \frac{(q)}{N}.$$

Распределение степеней. Не все вершины сети имеют одинаковое количество ребер. Распределение количества ребер вершины, то есть степень вершины, характеризуется функцией $P(< k >)$, которая определяет вероятность того, что случайно выбранная вершина будет иметь ровно k ребер. Поскольку в случайном графе ребра распределяются случайным образом, большая часть вершин имеет приблизительно одинаковую степень, близкую к средней степени $< k >$ сети.

Эти три концепции (малая длина пути, кластеризация, степень без масштабирования) привели к раздору в моделировании сетей в последние несколько лет, дав жизнь трем основным классам парадигм моделирования. Во-первых, случайные графы, являющиеся вариантом модели Эрдёша-Реньи, до сих пор используются во многих областях и являются основой для моделирования и эмпирических учений. Во-вторых, сразу после формулировки кластеризации, появился класс моделей, в целом называемых моделями малого мира. Эти модели представляют собой нечто среднее между высоко фрагментированными регулярными решетками и случайными графами. Наконец, открытие степенного распределения степеней вершин привело к появлению различных моделей без масштабирования, которые, концентрируясь на динамике сетей, должны объяснить происхождение степенного распределения степеней вершин и прочих отклонений от распределения Пуассона, имеющих место в реальных системах [4].

1.1.1. Случайные графы

Случайный граф - Теория случайных графов находится на стыке теории графов и теории вероятностей. Она вероятностная модель, предназначенная для изучения частотных характеристик различных параметров графов. Под случайным графом обычно понимается некий класс графов, на котором задано распределение вероятностей. Этот класс графов очень хорошо подходит для моделирования сетей связи, подверженных некоторым случайным изменениям, или логические сетей, элементы которых могут приходить в

неисправные состояния: при рассмотрении картины фазовых превращений в статистической физике; при изучении различных биологических процессов; при решении задач минимизации булевых функций и др.

Многие свойства таких случайных графов могут быть получены с помощью случайного анализа. С этой точки зрения Эрдёш и Реньи определили, что каждый граф обладает свойством Q, если при N → ∞, вероятность выполнения Q равна 1. Среди вопросов, рассмотренных Эрдёшем и Реньи, некоторые имеют прямое отношение к сложным сетям. Например, такие: Является ли стандартный граф связным? Содержится ли в нем треугольник из соединенных вершин? Каким образом диаметр зависит от размеров графа?

1.1.2. Модель Эрдеша–Реньи и его свойства

В теории графов модель Эрдеша – Реньи (ER) представляет собой модель для генерации случайного графа. Случайный граф определяется как N помеченных вершин, соединенных n ребрами, которые выбираются случайным образом из $\frac{N\cdot(N-1)}{2}$ возможных. Всего существует $\left(\frac{n}{\frac{N*(N-1)}{2}}\right)$ графов с N вершинами и n ребрами, которые образуют вероятностное пространство с равной вероятностью для каждой реализации.

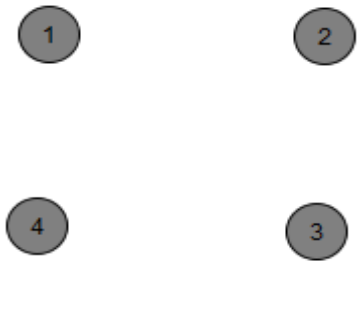
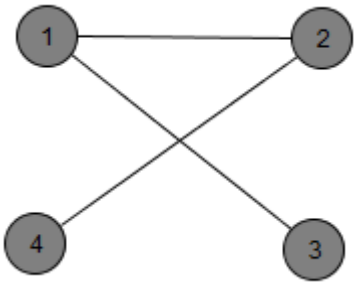
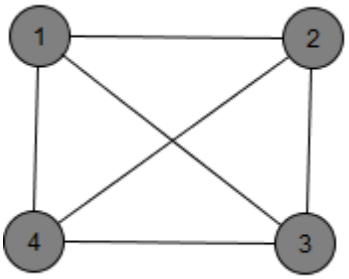
Согласно данной модели, граф конструируется с помощью соединения пар вершин случайным образом. Каждое ребро включается в граф с вероятностью p, независимо для каждой пары вершин. С ростом p от 0 до 1 граф становится все более насыщенным ребрами. При p = 0 имеем пустой граф (граф, не имеющий ребер), при $p = 1$– полный граф. При фиксированном количестве вершин в графе вероятность p – единственный входной параметр модели Эрдеша – Реньи. Очевидно что если G_0 — граф с N вершинами и n ребрами, то вероятность получить его с помощью этого процесса составит:

$$P(G_0) = p^n \cdot (1-p)^{\left(\frac{N\cdot(N-1)}{2}-n\right)},$$

а вероятность получения сети G'состоящий из N вершин и n ребер будет:

$$P(G') = \left(\frac{n}{\frac{N\cdot(N-1)}{2}}\right)\cdot p^n \cdot (1-p)^{\left(\frac{N\cdot(N-1)}{2}-n\right)}.$$

Таб. 3: Генерация ER графа для N = 1024, p={0.0, 0.5, 1.0}

		
N=4 p=0.0	*N=4 p=0.5*	*N=4 p=1.0*

Рассмотрим некоторые свойства сети Эрдеша-Реньи:

Свойства 1: Пусть имеем ER граф $G(N,p)$ и $p = \frac{c \cdot lnN}{N}$. В этом случае, если c>1, то почти всегда случайный граф связен, если c < 1, то почти всегда не является связным.

Свойства 2: Пусть имеем ER граф $G(N,p)$ и $p = \frac{c \cdot lnN}{N}$. Если c>3, то при N>100

$$P_{N,p}(G) \geq 1 - \frac{1}{N}.$$

От всего этого следует, что мы имеем резкий скачок от ‘почти всегда связност’ к ‘почти всегда несвязности’ и функция $\frac{c \cdot lnN}{N}$ служит границей перехода от ненадежности к надежности, а такой переход называют фазовым, а функцию p по пороговой.

Свойства 3: Пусть имеем ER граф $G(N,p)$ и $p = \frac{c}{N}$. Если c < 1, то существует такая константа $\alpha(c)$, что почти всегда размер каждой связной компоненты графа $< \alpha \cdot \ln N$. А если $c > 1$, то существует такая константа β(c), что почти всегда в графе есть ровно одна компонента размера $\geq \beta \cdot N$.

Свойства 4: Пусть p = $\frac{\ln N + c + O(1)}{N}$. Тогда $P_{N,p}(G) \to e^{-c}$, N→∞. В частности, при p=$\frac{\ln N}{N}$ вероятность стремится к e^{-1}. От этого следует, что здесь тоже асимптотическая вероятность связности есть, но она лежит на строгих пределах от 0 до 1.

Глава 2. РАСПРОСТРАНЕНИЕ АКТИВНОСТИ В СЕТИ

Для хорошего понимания что такое активность сети и нейрона надо понимать что из себя представляет искусственный нейронный сеть.

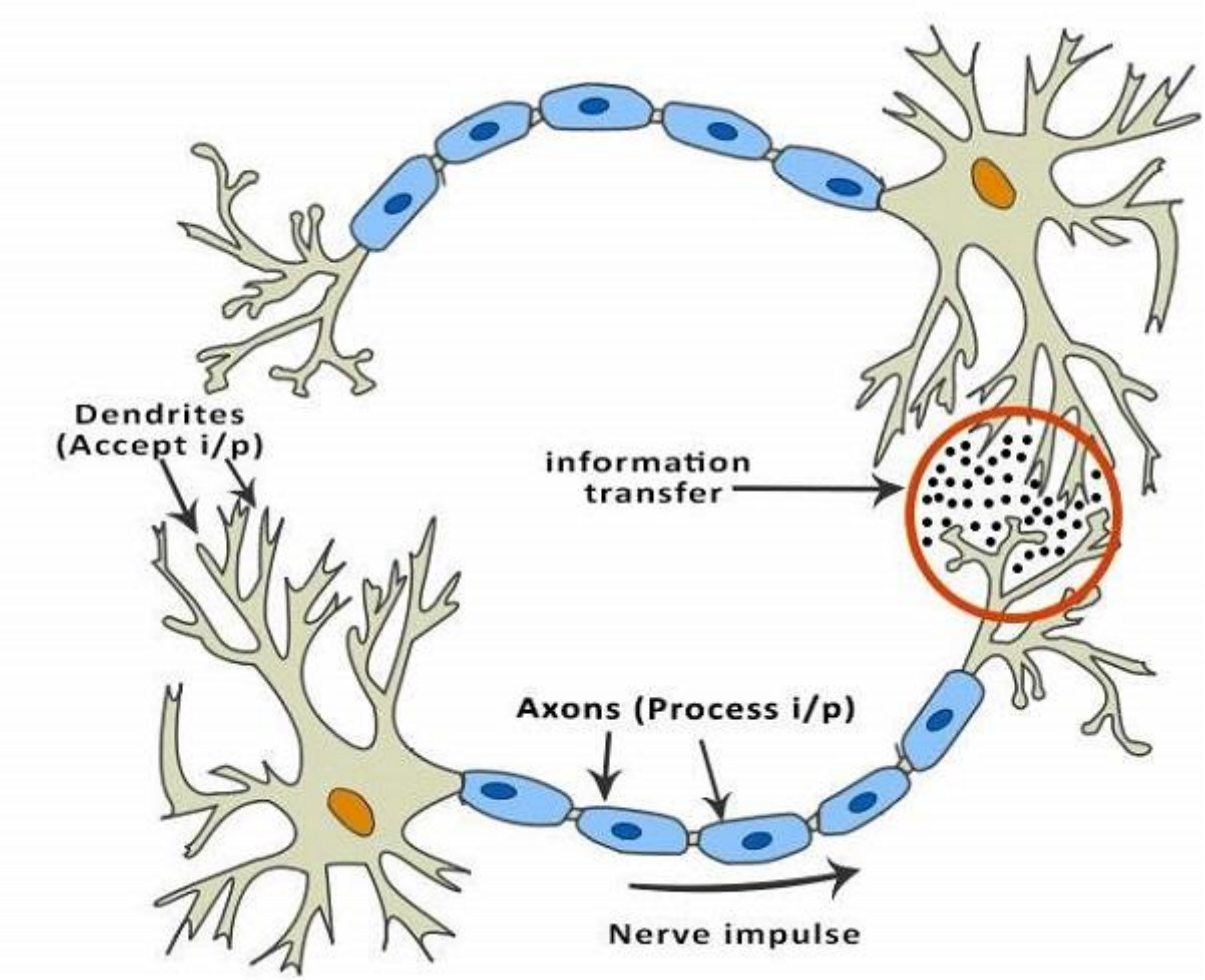


Рис. 2: Компоненты нейрона и как они взаимодействуют друг с другом.

Например человеческий мозг состоит из 86 миллиардов нервных клеток, называемых нейронами. Они соединены с тысячи другими клетками с помощью аксонам. Аксон — это нейрит (длинный цилиндрический отросток нервной клетки), по которому нервные импульсы идут от тела клетки (сомы) к иннервируемым органам и другим нервным клеткам.

Дендрит — разветвлённый отросток нейрона, который получает информацию через химические или электрические синапсы от аксонов других нейронов и передаёт её через электрический сигнал телу нейрона, из которого вырастает. Каждый нейрон состоит из одного аксона, тела и нескольких дендритов, в зависимости от числа которых нервные клетки делятся на униполярные, биполярные или мультиполярные. Передача нервного импульса происходит от дендритов (или от тела клетки) к аксону, а затем сгенерированный потенциал действия от начального сегмента аксона передаётся назад к дендритам.

Аналогично модели человеческого мозга в искусственной нейронной сети тоже клетки(нейроны) взаимодействуют друг с другом. В случае активности они могут передать импульс своих соседей пытаясь активировать неактивных (если такие существуют) и сами деактивироваться после передачи.

2.1. Описание процесса распространения активности

Пусть имеется сеть, состоящая из N нейронов, каждый из которых может находится в одном из двух состояний: активный или пассивный. Пусть m из N нейронов находятся в активном состоянии. Соседние нейроны взаимодействуют друг с другом с некой закономерностью – активные нейроны передают импульс соседним нейронам, пытаясь активировать пассивных и начиная с какого-то момента сами становятся пассивными. Все взаимодействия в сети прекращается только в случае, если в сети отсутствуют активные нейроны, т.е. когда сеть полностью затухает. Начиная с какого-то момента состояние сети может стать стабильном, т.е. плотность активности (отношение количеств активных - m и всех узлов - N) - ρ не меняется, несмотря на то, что между соседними нейронами продолжается взаимодействие:

$$\rho = \frac{m}{N} = const.$$

2.2. Разные подходы для распространения активности

С первого взгляда вышеприведенное описание кажется простым и понятным, но пытаясь глубоко описать, реализовать и исследовать характерные свойства вышеприведенного процесса, возникают ряд вопросов, в частности какова закономерность активации-деактивации нейрона, передачи импульса активного нейрона другим, что из себя представляет один шаг всего процесса, после скольких шагов или начиная с какого состояния можно полагать, что текущий процесс стал стабильным(т.е плотность активности практически не меняется) и вся картина процесса уже понятно.

Следует отметить, что меняя характеристику подходов для каждого из упомянутых вопросов может привести к полному изменению всего процесса и для понимания как преимуществ, так и недостатке каждого подхода надо сделать эксперименты почти для каждого из них.

В этой работе рассмотрены три основные подходы, детали и результаты каждого из которых будет подробно описан.

Начнем из формальных описаний алгоритмов:

Пусть имеется сеть (заданная с помощью матрицы смежности, списка смежности или списка рёбер), дан список начальных активных вершин, значения μ – параметр деактивации, λ – параметр активации, max_step_count - максимальное количество шагов.

Описание алгоритма 'A'(по всему списку):

Вход: Сеть, список активных узлов, μ, λ, max_step_count**.**

*Шаг 1***:** Построить представление заданной сети, отметить начальные активные узлы.

*Шаг 2***:** Если в сети отсутствуют активные вершины или количество шагов больше данного максимального значения п - перейти к шагу *Выход***.**

*Шаг 3***:** Случайным образом (с равной вероятностью) выбрать один узел из всего списка вершин – n_1**.**

Шаг 4**:** Если выбранный узел пассивный - перейти к шагу 3. В противном случае перейти к шагу 5.

*Шаг 5***:** Случайным образом (с равной вероятностью) выбрать один из его соседей – n_2. Если узел n_2 пассивен, то активизировать выбранный узел с вероятностью $\frac{\lambda}{(\lambda+\mu)}$, а в случае активности перейти к шагу 6**.**

*Шаг 6***:** С вероятностью $\frac{\mu}{(\lambda+\mu)}$ деактивировать начальный выбранный узел n_1**.**

*Шаг 7***:** Переход к *шагу 2*.

*Выход***:** Список значений ρ (плотность активных узлов) после каждого шага.

Описание алгоритма 'B' (по соседным списком):

Вход: Сеть, список активных узлов, μ, λ, max_step_count**.**

*Шаг 1***:** Построить представление данного сети, отметить заданные вершины активными.

*Шаг 2***:** Если сеть полностью неактивный (в сети отсутствуют активные вершины) или количество шагов больше заданного максимального значения - перейти к шагу *Выход*.

*Шаг 3***:** Случайным образом (с равной вероятностью) выбрать один узел из всего списка – n_1.

Шаг 4**:** Если выбранный узел пассивный - перейти к шагу 3. В противном случае перейти к шагу 5.

*Шаг 5***:** Для каждого соседнего узла в случае пассивности активировать с вероятностью $\frac{\lambda}{(\lambda+\mu)}$**.**

*Шаг 6***:** С вероятностью $\frac{\mu}{(\lambda+\mu)}$ деактивировать изначально выбранный узел n_1**.**

*Шаг 7***:** Переход к *шагу 2*.

*Выход***:** Список значений ρ после каждого шага.

Описание алгоритма 'C'(по списку активных, с пересчетом времени):

Вход: Сеть, список активных узлов, μ, λ, max_step_count**.**

*Шаг 1***:** Построить представление данной сети, отметить активные начальные вершины

Шаг 2: Если сеть полностью неактивный (в сети отсутствуют активные вершины) или количество шагов больше данного максимального значения - перейти к шагу *Выход*.

Шаг 3: Построить список активных узлов List присутствующий в последнем Шаг-2 и для каждого узла n_1 из этого списка

Шаг 3.1: Случайным образом (с равной вероятностью) выбрать один из соседних узлов n_2.

Шаг 3.1.1: В случае пассивности активировать n_2 с вероятностью $\frac{\lambda}{(\lambda+\mu)}$.

Шаг 3.2: С вероятностью $\frac{\mu}{(\lambda+\mu)}$ деактивировать текущий n_1 узел.

Шаг 4: Переход к *шагу 2*.

Выход: Список значениев ρ после каждого шага.

2.3. Тестирование алгоритмов распространения активности в сети.

Как и на многих сферах науки в этом случае тоже каждое исследование, процесс требует четкое тестирование и доказательство корректности рекомендуемого алгоритма.

Один из принципов тестирования является рассмотрение поведения алгоритма в критических ситуациях, где каждое неправильное поведения будут ясно выражено. Здесь будем исследовать:

1) Процесс распространения активности в сети, в котором включено только активизация.

2) Процесс распространения активности в сети, в котором включено только деактивация.

Для первого случая необходимо и достаточно чтобы $\mu = 0$, а для второго $\lambda = 0$. Но в обоих случаях диапазон исследования слишком ограничен: в первом случае из $\mu = 0$ следует, что $\lambda/(\lambda + \mu) = 1 \quad \forall \lambda \in (0,1]$, который означает, что результат не зависит от λ. То же самое имеется место и для второго случая

$\lambda = 0 \rightarrow \mu/(\lambda + \mu) = 1 \quad \forall \mu \in (0,1]$, т.е поведение процессов не зависит от значений μ и λ соответственно.

Для обхода вышеприведенных ограничении в исследовании заменяется

$\lambda/(\lambda + \mu)$ на λ, а $\mu/(\lambda + \mu)$ на μ.

В *приложении 1* представлены результаты процесса распространения активности для сети состоящий из 1024 вершин, с использованием алгоритма ‘C’, где включено только а) активация, б) деактивация. Из результатов видно, что весь процесс происходит так, как и ожидалось. В частном случае для $\{\mu = 1.0, \lambda = 0.0\}$ вся сеть затухает после первого же шага, так как на первом шаге все вершины(так как все вершины в активном состоянии)

деактивируются с равной вероятностью. А дальше уменьшение μ приводит к увеличению шагов процесса, т.к. вероятность деактивации текущего нейрона уменьшается.

Надо иметь в виду, что в случае $\mu = 0.0$ стационарным является состояние, когда плотность активности равно единицы, т.к. в этом случае во время процесса в сети не будет деактивация и текущее состояние длиться вечно.

Глава 3. ЭТАПЫ ПРОЦЕССА ИССЛЕДОВАНИЯ

Анализ процесса распространения активности в сети состоит из этапов, каждое из которых надо моделировать очень тщательно. Ниже приведены главные концепции каждого этапа.

3.1. Генерация сетей

В начале, в случае отсутствия, генерируется сеть. На рамках этой работы используются случайные сети, в частности модель Эрдеша – Рень**.** Это означает что если сеть имеет N вершин, компонент связности - p , то количество ребер по всей вероятности будет $p \cdot \frac{N(N-1)}{2}$**,** а степень случайно выбранной вершины $(N-1) \cdot p$.

3.2. Структурный анализ полученных сетей

Для понимания основных характеристик данной сети (в частности количество существующих кластеров в сети) делается структурный анализ сети. Анализ включает:

1) Представление спектра собственных значений матрицы.
2) Распределение собственных значений матрицы.

3.3. Процесс активации.

Это основной этап всего процесса, который включает: сам процесс распространения активности. Так как здесь мы имеем дело с вероятностным пространством, все эксперименты с тем же самыми начальными данными надо делать на нескольких экземплярах. В рамках нашего исследования в качестве усредненного результата выбирается самый часто встречающийся. Но есть и другой подход для получения более реалистичную картину всего процесса, которое пока что не использовалось:

Пусть имеется n экземпляров $\{c_1, c_2, \dots, c_n\}$, с теми же самыми начальными данными, где c_i количество шагов в i –ом экземпляре.

Процесс усреднения вычисляется следующим образом:

1) Найти максимальное значение : $C = \max\{c_i\} / \forall i \in [1, n]$.
2) Для всех экземпляров продолжить процесс до *C* шага.
3) Так как на данном этапе c_i=*C* / $\forall i \in [1,n]$, построить усредненный результат:

   $V = \{v_1, v_2, \dots, v_n\}$, $v_i = \sum \rho_{ij}$ /$i \in [1, n]$, $j \in [1, c]$, где ρ_{ij} – плотность активности в *j*-ом шаге *i*-ого экземпляра.

А для нахождения фазы неустойчивого горения можно зафиксировать значение $\mu'(\lambda)$**,** и для $\lambda(\mu)$ найти значение $\lambda'(\mu')$, в случае которого $\lambda = \lambda' + \varepsilon$ сеть активируется, а в случае $\lambda = \lambda' - \varepsilon$ затухает.

3.5. Графическое представление процесса распространения активности

Как и на многих других случаях здесь тоже графическое представление всего процесса иногда дает как визуальное так и гораздо глубокое понимание всего процесса. Пример визуализации приведен Приложении 2.

В процессе активации красным цветом отмечаются активные, а желтым – пассивные вершины.

Цвет ребер также меняется:

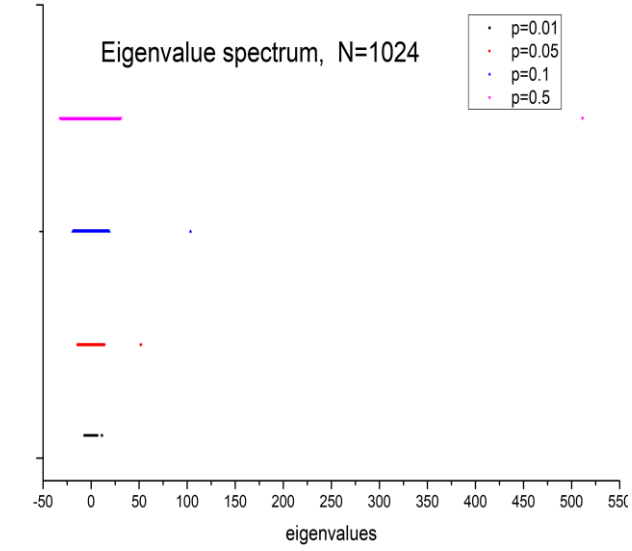
- красный – оба компоненты в активном состоянии,
- зеленый – на текущем шаге активность передается соседнему узлу,
- черный - в остальных случаях.

Глава 4. РЕЗУЛЬТАТЫ АНАЛИЗА

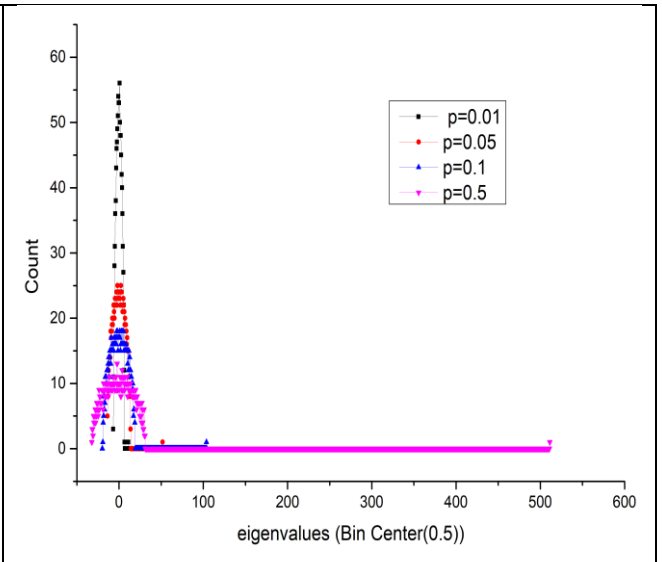
В этом разделе представлены результаты процесса с использованием алгоритмов описанные в главе 2.2. Так как во всех случаях используются те же самые сети, перед анализом результатов представим структурный анализ полученных сетей.

В качество сети используется модель Эрдеша – Реньи. Сеть состоит из $N = 1024$ вершин, $p = \{0.01, 0.05, 0.1, 0.5\}$, в начале в сети имеется 98 активных вершин(~0.1%).

Таб. 4: Структурный анализ для ER сети с параметрами $\{n = 1024, p = \{0.01, 0.05, 0.1, 0.5\}\}$



Спектр собственных значений



Распределение спектра собственных значений

Из вышеприведенных графиков следует что во всех случаях в сети существует только один кластер, а коэффициент кластеризации обратно пропорционально к значению p.

4.1. Исследование алгоритма ‘A’ для случайных сетей (по всему списку)

Рассмотрим поведение процесса для $N = 1024$, $p = \{0.5, 0.1, 0.05\}$, $\mu = 0.0$

A) $p = 0.5$

Таб. 5: Количество шагов для алгоритма ‘A’ , p = 0.5.

λ	**Steps count**
0.5	14911
0.1	79371
0.05	186588
0.01	500000+
0.005	500000+
0.001	500000+
0.0005	500000+
0.0001	500000+
0.00005	500000+

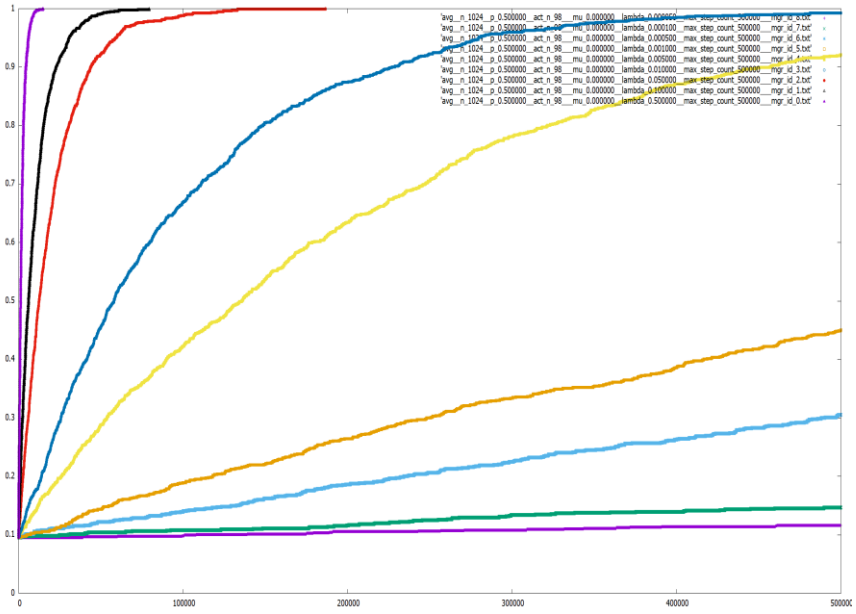


Рис. 3: Результаты для **p=0.5, μ = 0.0, λ={0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005}**

Б) $p = 0.1$

Таб. 6: Количество шагов для алгоритма 'A', p = 0.1.

λ	**Steps count**
0.5	14323
0.1	71546
0.05	130049
0.01	500000+
0.005	500000+
0.001	500000+
0.0005	500000+
0.0001	500000+
0.00005	500000+

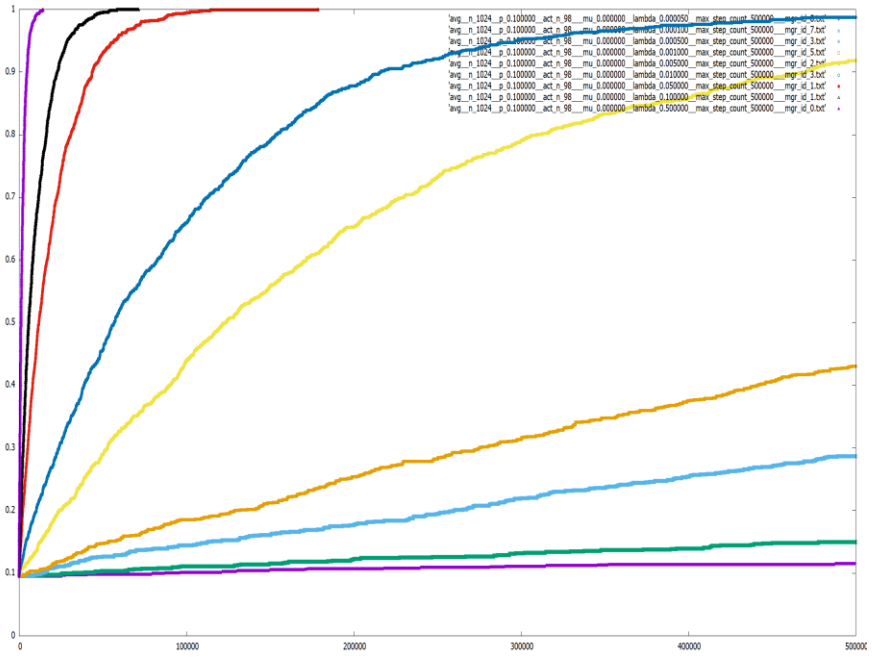


Рис. 4: Результаты для **p=0.1, μ = 0.0, λ={0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005 }**

B) $p = 0.05$

Таб. 7: Количество шагов для алгоритма 'A' , p = 0.05.

λ	**Steps count**
0.5	18389
0.1	95531
0.05	184266
0.01	500000+
0.005	500000+
0.001	500000+
0.0005	500000+
0.0001	500000+
0.00005	500000+

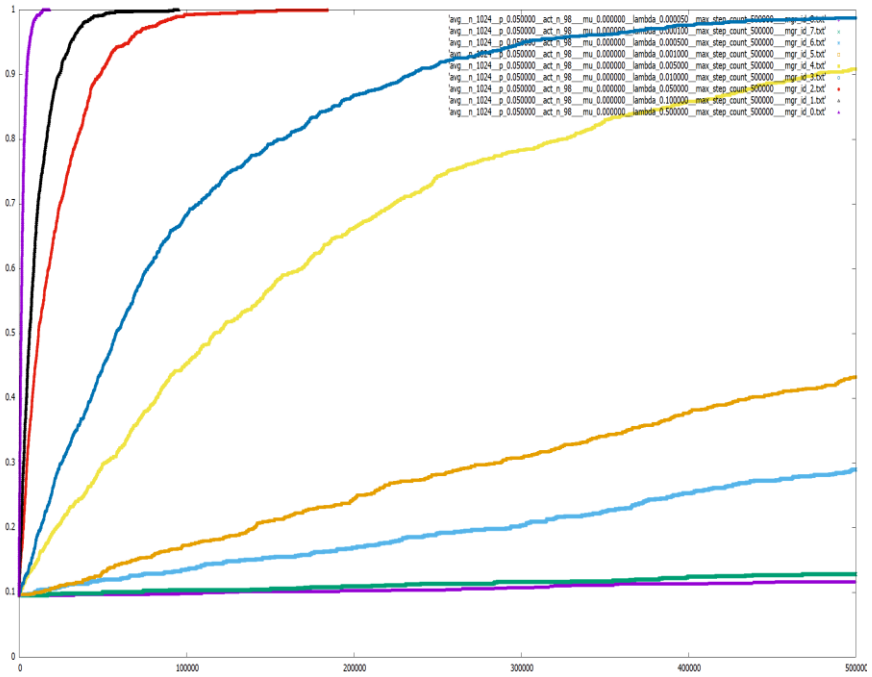


Рис. 5: Результаты для **p=0.1, μ = 0.0, λ = { 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005}**

От вышеприведенного графика следует, что скорость распространения активности прямо пропорциональна к значению **λ и** распространение активности в сети в конце всего процесса происходит очень медленно, т.к. вероятность выбора пассивного соседнего узла значительно уменьшается.

Так как скорость распространения активности прямо пропорциональна к значению **λ**, возникает вопрос: существует ли закономерность между скоростью распространения активности в сети и **p**. Для визуального представления сравним полученные траектории:

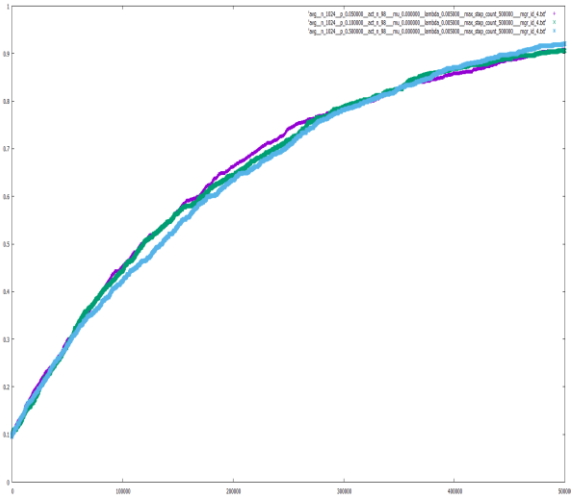


Рис. 6: P={0.5, 0.1, 0.05} λ = 0.005

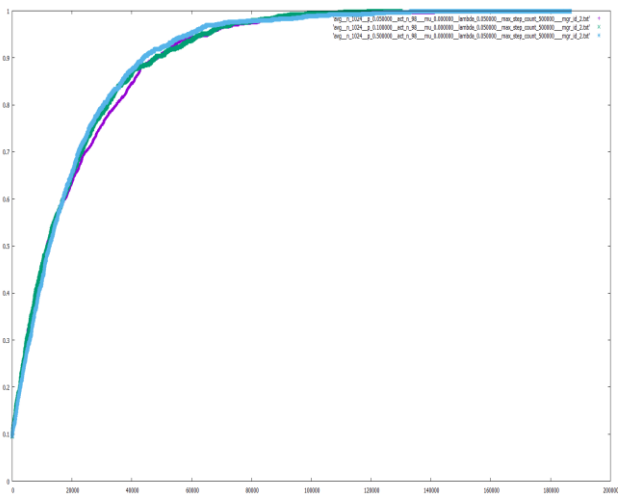


Рис. 7: P={0.5, 0.1, 0.05} λ = 0.05

Как видно из вышеприведенных графиков поведение процесса одинакова для сетей с разными значениями компонентов связности. Такой результат тоже ожидаемый, т.к.

связность как ‘помогает’(возрастая вероятность, что вершина будет иметь пассивного соседа), так и ‘мешает'(уменьшая вероятность, что случайно выбранный соседний узел будет пассивным).

4.2. Исследование алгоритма ‘B’ для случайных сетей (по соседным списком)

Рассмотрим поведение процесса для $N = 1024$, $p = \{0.5, 0.1, 0.05\}$, $\mu = 0.0$

А) $p = 0.5$

Таб. 8: Количество шагов для алгоритма ‘B’ , p = 0.5.

λ	**Steps count**
0.5	29
0.1	243
0.05	274
0.01	1510
0.005	2960
0.001	13361
0.0005	29313
0.0001	151329
0.00005	279031

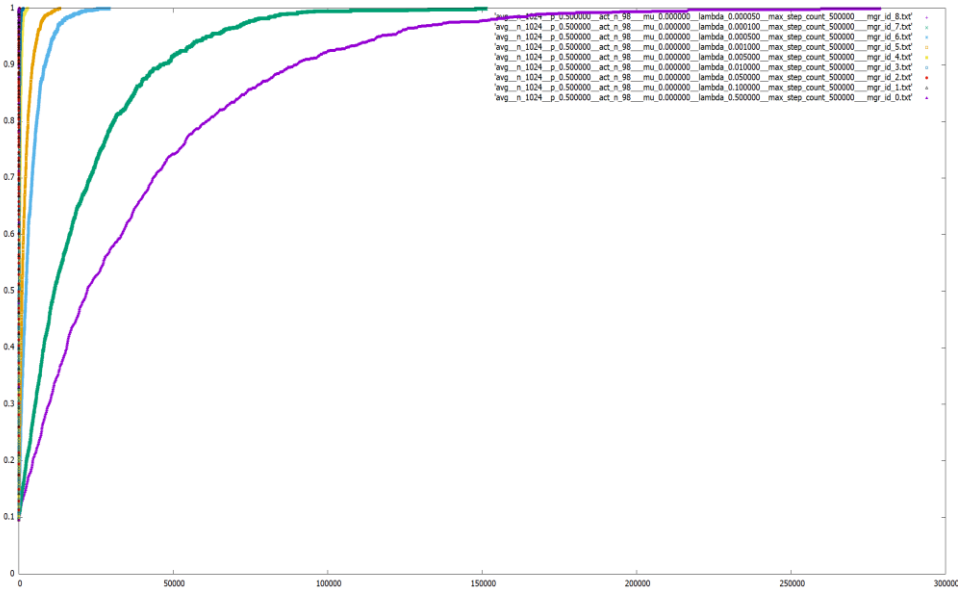


Рис. 8: Результаты для **p=0.5, μ = 0.0, λ = { 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005}**

Б) $p = 0.1$

Таб. 9: Количество шагов для алгоритма ‘B’ , p = 0.1.

λ	**Steps count**
0.5	188
0.1	830
0.05	1459
0.01	6961

0.005	18675
0.001	83304
0.0005	160886
0.0001	500000+
0.00005	500000+

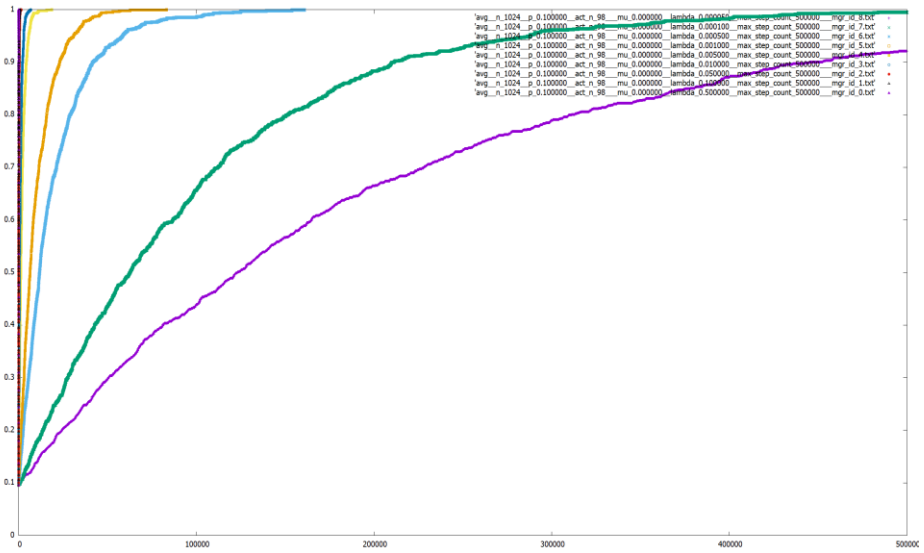


Рис. 9: Результаты для **p=0.1, μ = 0.0, λ = { 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005}**

B) $p = 0.05$

Таб. 10: Количество шагов для алгоритма 'B', p = 0.05.

λ	**Steps count**
0.5	313
0.1	1727
0.05	3425
0.01	13012
0.005	53180
0.001	159883
0.0005	297850
0.0001	500000+
0.00005	500000+

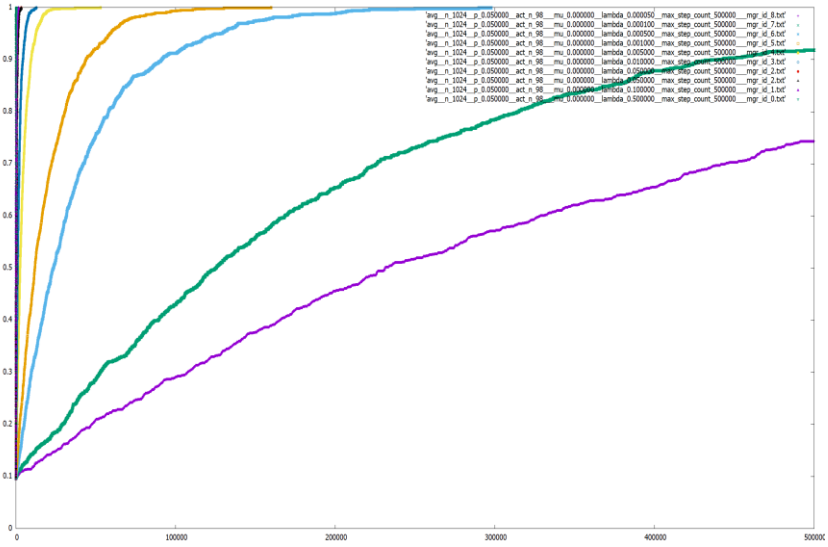


Рис. 10: Результаты для **p=0.05, μ = 0.0, λ = { 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005}**

Как и ожидалось на этом случае тоже скорость распространения активности прямо пропорциональна к значении **λ.**

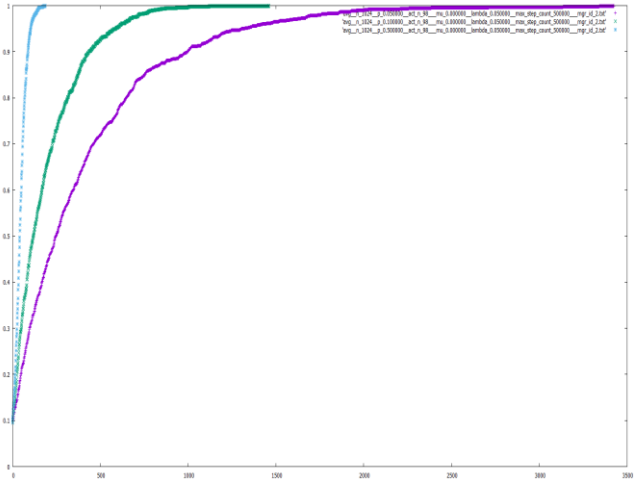


Рис. 11: P = {0.5, 0.1, 0.05}, λ = 0.05

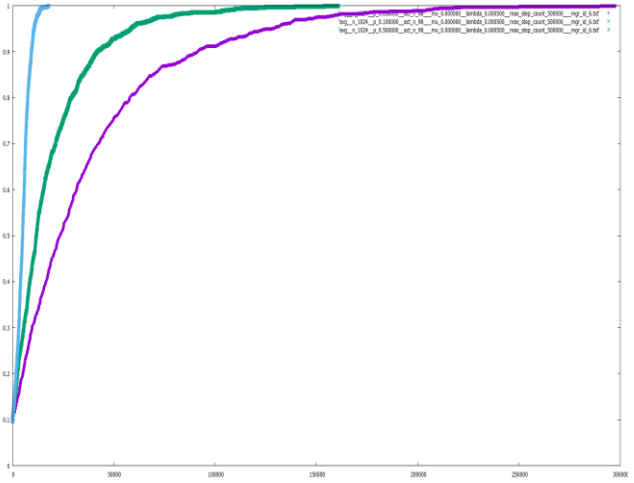


Рис. 12: P = {0.5, 0.1, 0.05}, λ = 0.0005

Как видно из полученных результатов, в случае алгоритма ’B’ уменьшение p влияет на поведению процесса - уменьшая скорость распространения активности. А это связано с тем, что на каждом шаге текущий узел пытается активировать всех своих пассивных соседей.

Для хорошего понимания поведенческих разниц алгоритмов ‘A’ и ‘B’ оценим один из критических случаях процесса: пусть ER(p) сеть состоит из n вершин и в начале процесса в сети имеется только одна активная вершина:

Таб. 11: Оценка вероятности. На первом шаге текущий нейрон активирует как минимум одного соседа для алгоритмов ‘A’ и ‘B’:

‘A’	‘Б’
$\frac{1}{n} \cdot \lambda$	$\frac{1}{n} \cdot (1 - (1 - \lambda)^{p \cdot (n-1)})$

Таб. 12: Оценка вероятности. На первом шаге текущий нейрон активирует m соседних нейронов (1<m<=p·(n-1)) для алгоритмов ‘A’ и ‘B’:

‘A’	‘Б’
0	$\frac{1}{n} \cdot (\binom{m}{p \cdot (n-1)} \cdot \lambda^m \cdot (1 - \lambda)^{(p \cdot (n-1) - m)})$

Таб. 13: Оценка вероятности. На первом шаге текущий нейрон не активирует не одного соседа для алгоритмов ‘A’ и ‘B’:

‘A’	‘Б’
$\frac{n-1}{n} + \frac{1}{n} \cdot (1 - \lambda)$	$\frac{n-1}{n} + \frac{1}{n} \cdot (1 - \lambda)^{p \cdot (n-1)}$

Таб. 14: Оценка вероятности. На первом шаге сеть будет затухать для алгоритмов ‘A’ и ‘B’:

‘A’	‘Б’
$\mu \cdot (\frac{1}{n} \cdot (1 - \lambda))$	$\mu \cdot (\frac{1}{n} \cdot (1 - \lambda)^{p \cdot (n-1)})$

4.3. Исследование алгоритма ‘C’ для случайных сетей (по списку с пересчетом времени)

Рассмотрим поведение процесса для $N = 1024$, $p = \{0.5, 0.1, 0.05\}$, $\mu = 0.0$

A) p = 0.5

Таб. 15: Количество шагов для алгоритма ‘B’, p = 0.5.

λ	**Steps count**
0.5	21
0.1	100

0.05	201
0.01	1069
0.005	2182
0.001	7672
0.0005	17422
0.0001	85396
0.00005	186001

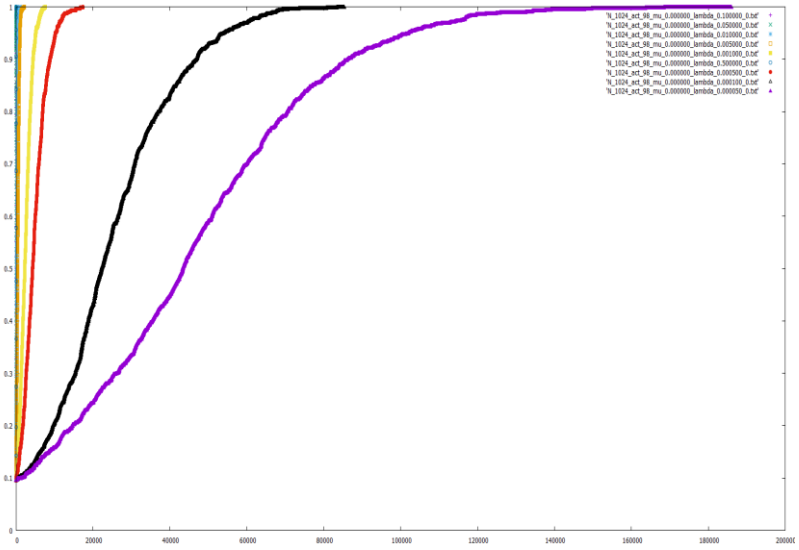


Рис. 13: Результаты для **p=0.5, μ = 0.0, λ = { 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005}**

Б) p = 0.1

Таб. 16: Количество шагов для алгоритма 'C' , p = 0.1.

λ	**Steps count**
0.5	26
0.1	143
0.05	216
0.01	887
0.005	1824
0.001	10526
0.0005	22147
0.0001	92783
0.00005	156089

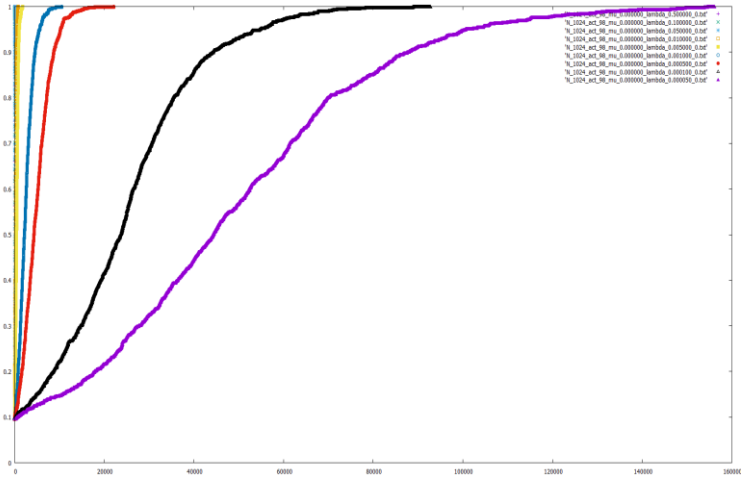


Рис. 14: Результаты для **p=0.1, μ = 0.0, λ = { 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005}**

B) p = 0.05

Таб. 17: Количество шагов для алгоритма 'C' , p = 0.05.

λ	**Steps count**
0.5	23
0.1	91
0.05	231
0.01	1147
0.005	2275
0.001	8104
0.0005	23146
0.0001	89546
0.00005	171129

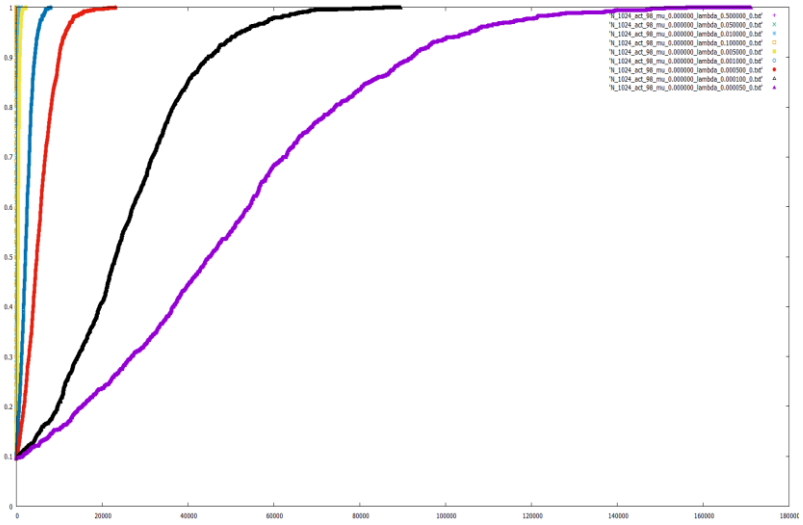


Рис. 15: Результаты для **p=0.5, μ = 0.0, λ = { 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005}**

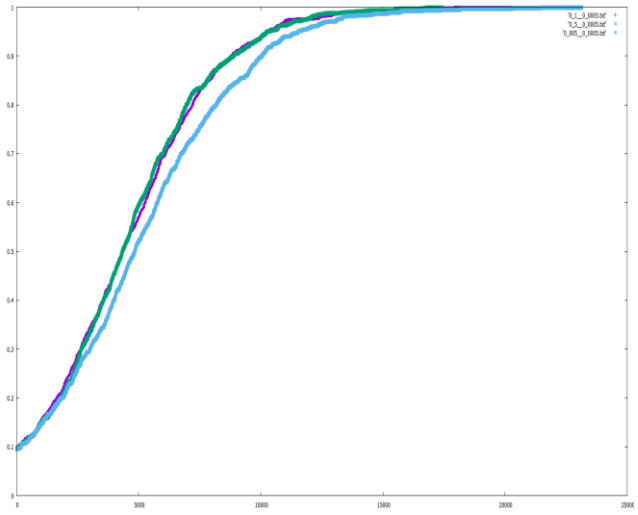


Рис. 16: P = {0.5, 0.1, 0.05}, λ = 0.0005

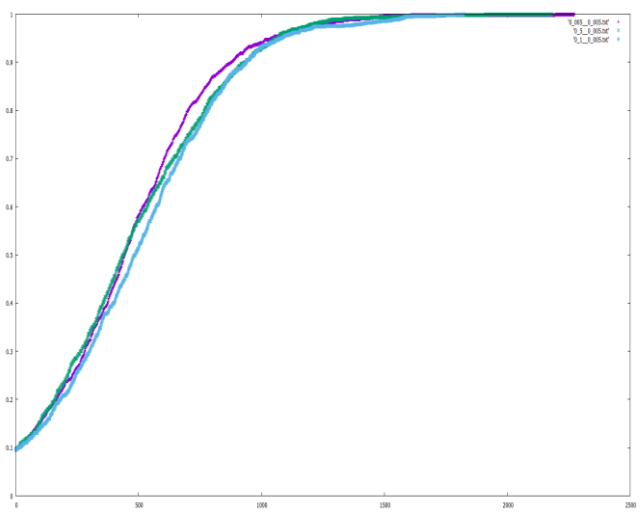


Рис. 17: P = {0.5, 0.1, 0.05}, λ = 0.0005

Как и в случае алгоритма ‘A’, здесь тоже скорость распространения активности не зависит от значения p. Чтобы получить зависимость надо вместо случайного выбора одного соседнего рассмотреть все соседние нейроны и активировать каждый с вероятностью λ.

А теперь оценим поведение процесса: пусть E-R(p) сеть состоит из n вершин и в начале процесса в сети имеется только одна активная вершина:

Таб. 18: Оценка вероятности. На первом шаге текущий нейрон активирует как минимум одного соседа для алгоритма 'B':

‘C’
λ

вероятность, что на первом шаге текущий нейрон активирует m соседних нейронов будет $(1 < m <= p \cdot (n-1))$:

Таб. 19: Оценка вероятности. На первом шаге текущий нейрон активирует m соседних нейронов $(1 < m <= p \cdot (n-1))$ для алгоритма 'B':

‘C’
0

Таб. 20: Оценка вероятности. На первом шаге текущий нейрон не активирует не одного соседа для алгоритма 'B':

‘C’
$(1-\lambda)$

Таб. 21: Оценка вероятности. На первом шаге сеть будет затухнуть для алгоритма 'B':

‘C’
$\mu \cdot (1-\lambda)$

Глава 5. ТРЕБОВАНИЕ И ЭТАПЫ РАЗРАБОТКИ ПРОГРАММНОЙ СИСТЕМЫ

Так как весь процесс распространения активности – генерация случайной сети, выбор нейронов, передача активности, деактивация нейрона, основано на теории вероятностей, то для получения точных результатов надо тот же исследование с одинаковыми начальными данными делать много раз, т.е. чем больше количество экземпляров, тем большая точность результатов. Еще надо иметь в виду, что шаг, начиная которого плотность активности практически не меняется, может быть очень большим. Вот например если значение μ фиксировано, λ принимает 20 значения, количество экземпляров 10, максимальное количество шагов 10’000’000, то общая количества шагов для каждого алгоритма в худшем случае будет:

$$20 \cdot 10 \cdot 10'000'000 = 2'000'000'000.$$

Основные требования к разработке:

1) Корректность,
2) Эффективность,
3) Платформенная независимость.

В наличие таких требований, язык для разработки основного ПО был выбран C++.

Для повышения производительности в уровне системы сеть представляется с помощью соседним списком: std::vector<std::vector<id_type>>, где всем вершинам однозначно соответствует *id* в интервале *[0, N-1]*. Благодаря этому сложность времени получения случайного соседа для данного узла константа - *O(1)* (т.к. соответствующий промежуточный элемент получается с помощью перемещения базового указатель: base_ptr+idx). Другим важным аспектом является оптимальное проверка состояния данного узла, для чего используется std::vector<bool>, благодаря которому временная сложность проверки и обновление состояния тоже *O(1)*. Но т. к. перед каждым шагом надо проверить количество активных узлов в сети, еще хранится и счетчик, который обновляется во время каждой изменения состояния. Также во время всего процесса для получения плотность активности, вместо деления количество активных узлов на количеству всех узлов, вначале сохраняется соотношение *1/{N}*, и во время процесса количество активных узлов текущего шага умножается с этим значением.

Так как весь процесс исследование основано на теории вероятностей (генерация случайных чисел, равномерное распределение вероятностей и т. д.), один из важных концепций был корректное программное моделирование соответствующих формул, для чего использовались стандартные функции библиотеки <random>, в частности:

- $std :: random\_device$ – для генерации случайных чисел
- $std :: uniform\_distribution$ – для выбора текущей/соседних нейронов

♦ $std::bernoulli\_distribution$ – для активации/деактивации нейрона.

В качестве начальных данных передаются сеть(с помощью матрицы смежности, списка смежности или списка ребер), начальный список активных вершин, значения μ – скорость(вероятность) деактивации, λ – скорость(вероятность) активации, максимальное количество шагов, количество экземпляров. Вместо сети можно передать параметры для генерации случайной сети модели Эрдеша-Реньи: N – количество вершин, p - параметр вероятность связи. Сгенерированный сеть сохраняется в соответствующем файле.

После подготовительной работы (создании сети и инициализации соответствующих параметров) начинаются процессы распространения активности с использованием вышеупомянутых трех алгоритмов, для каждого из которых с *exp_count* экземплярами. Для каждого экземпляра сохраняются результаты соответствующих процессов (в частности состояние плотности активности после каждого шага). Как только все процессы заканчивают исполнение и все данные в готовом состоянии, происходит пересчет результатов для каждого алгоритма. Полученные результаты сохраняются в соответствующих файлах, после чего графически сравниваются результаты трех алгоритмов.

Так как нахождения фазы неустойчивого горения является один из важнейших вопросов процесса распространения активности в сети и в основном все исследования делаются на виртуальных машинах (с характеристиками 32cores, 252.2Gbytes), на языке Python написан скрипт, который в качестве начальных данных получает данные описанные выше, но вместо одного значения λ получает целый список $\{\lambda_1, \lambda_2, \dots, \lambda_k\}$, количество создаваемых процессов – proc_count и разделяя $\{\lambda_i\}$ множество на $[\frac{k}{proc\_count}]$ равные подмножество, создает proc_count процессов и вычисления для каждого $\{\lambda_{ij}\}$подмножество запускает на соответствующем процессе(используя основная программная средства написанная на C++).

Для визуализации процесса использовались стандартные структуры кроссплатформенного фреймворка Qt, в частности:

{*QGraphicsScene, QImage, QPainter, QBrush, QGraphicsEllipseItem, QgraphicsLineItem}.*

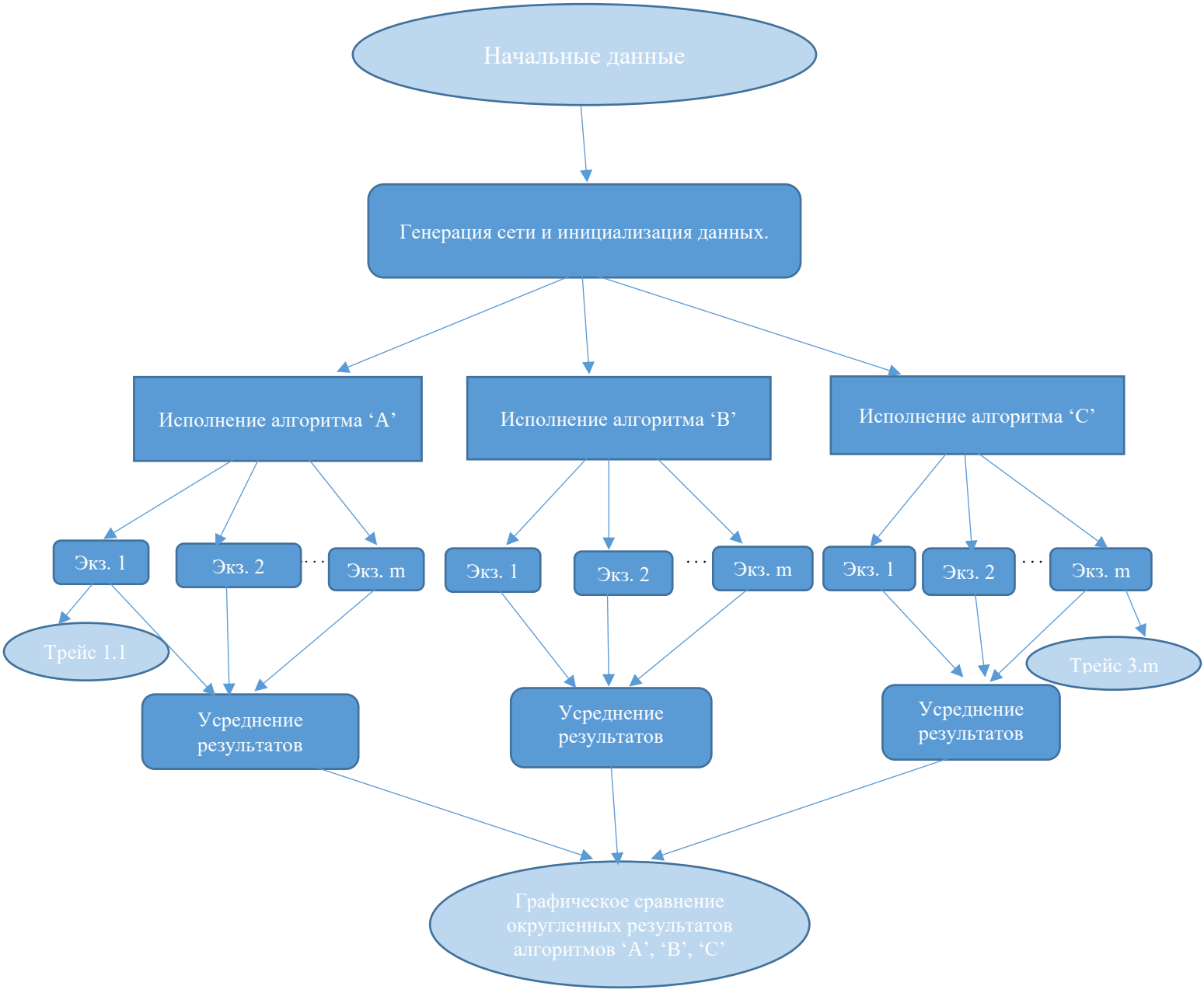
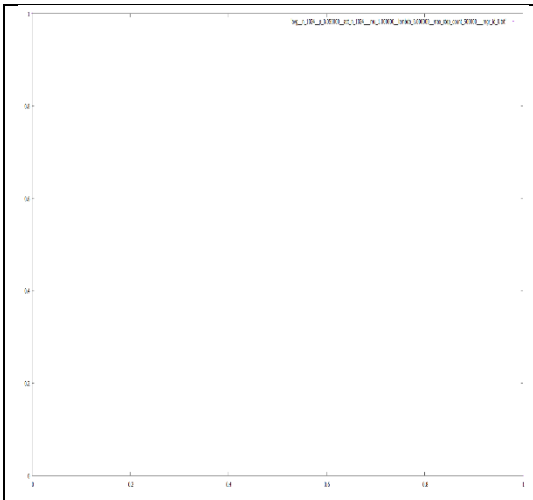


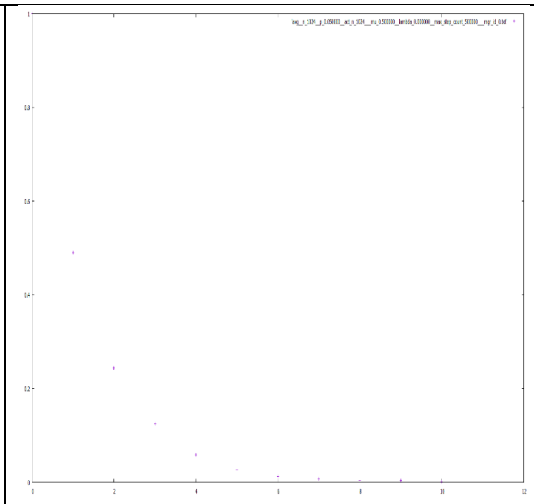
Рис. 18: Диаграмма этапов анализа.

Приложение 1. Графические представления результатов алгоритма ‘C’ для случаев, когда во время процесса включена только а) активация, б) деактивация.

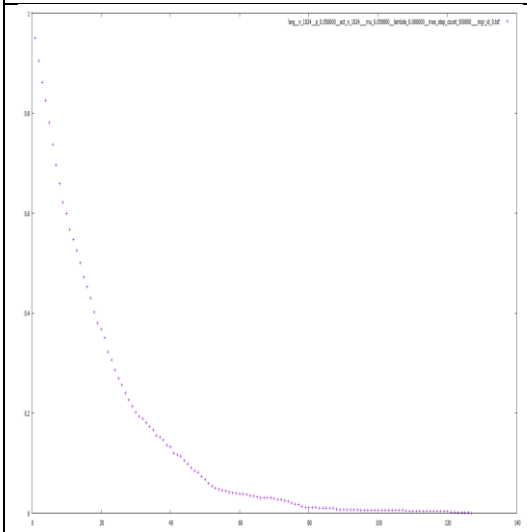
Таб. 23:Результаты алгоритма ‘C’ для случаев: μ ={0.1, 0.5, 0.05, 0.01}, λ=0



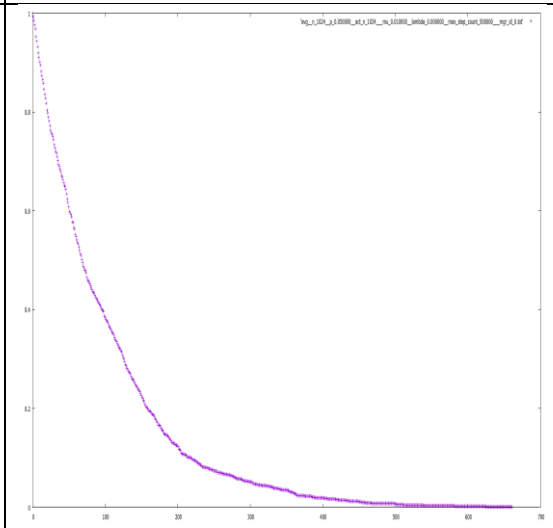
μ = 1.0, step_count = 1



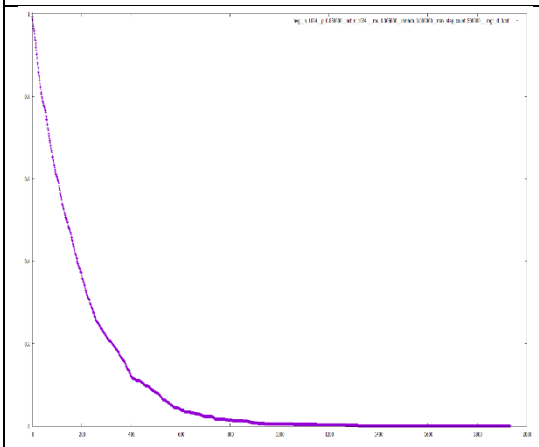
μ = 0.5, step_count = 11



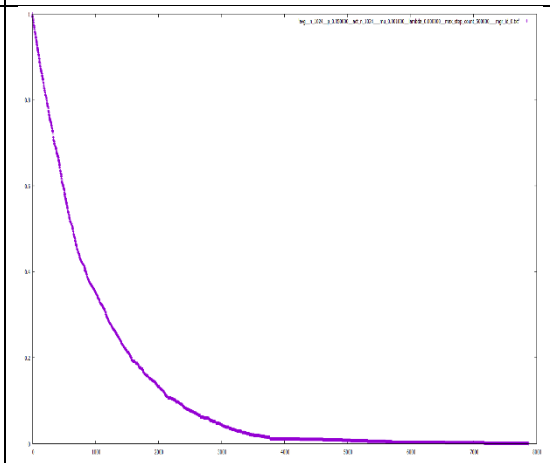
μ = 0.05, step_count = 127



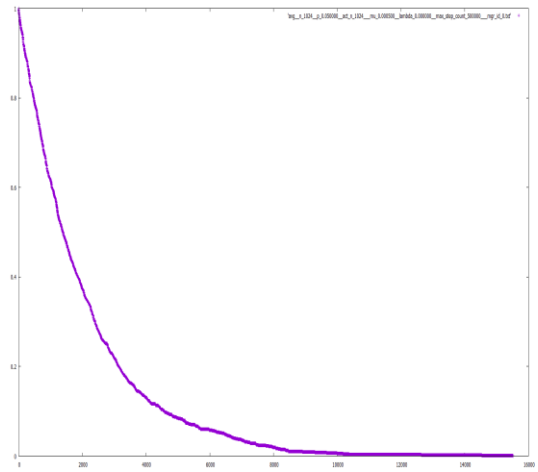
μ = 0.01, step_count = 661



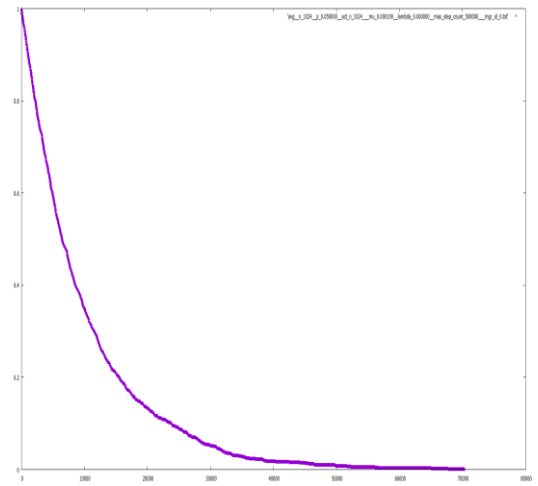
μ = 0.005, step_count = 1935



μ = 0.001, step_count = 7870

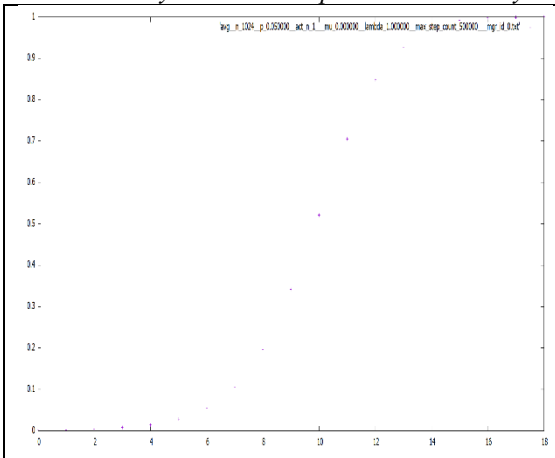


$\mu = 0.0005$, step_count = 15513

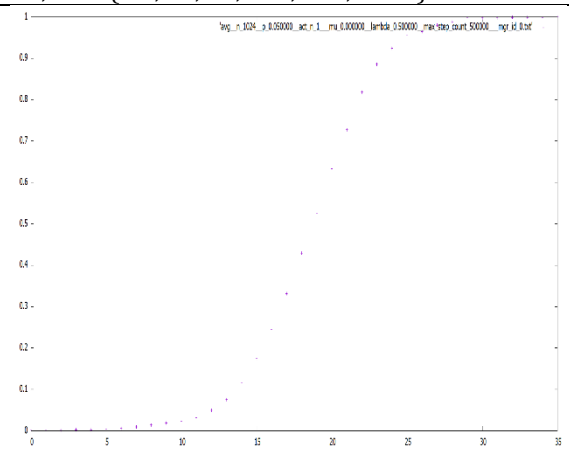


$\mu = 0.001$, step_count = 70349

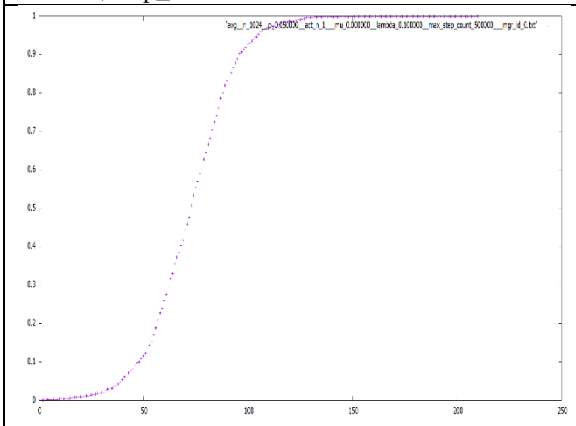
Таб. 24: Результаты алгоритма 'C' для случаев: $\mu = 0.0$, $\lambda = \{1.0,0.5,0.1,0.05,0.01,0.005\}$



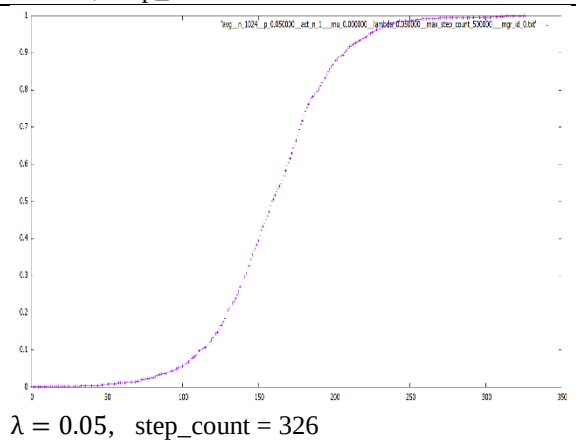
$\lambda = 1.0$, step_count = 18



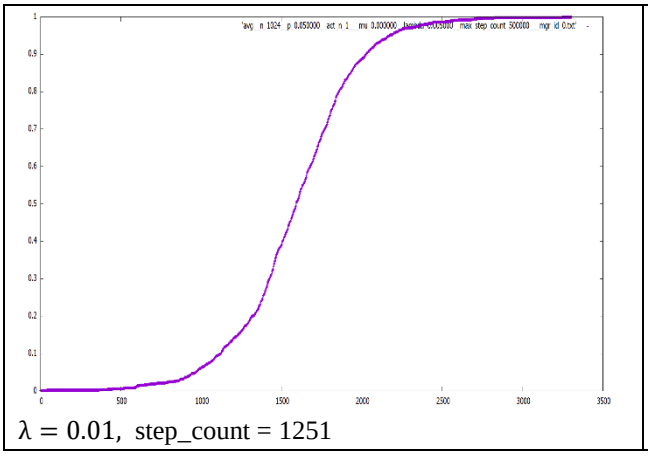
$\lambda = 0.5$, step_count = 35



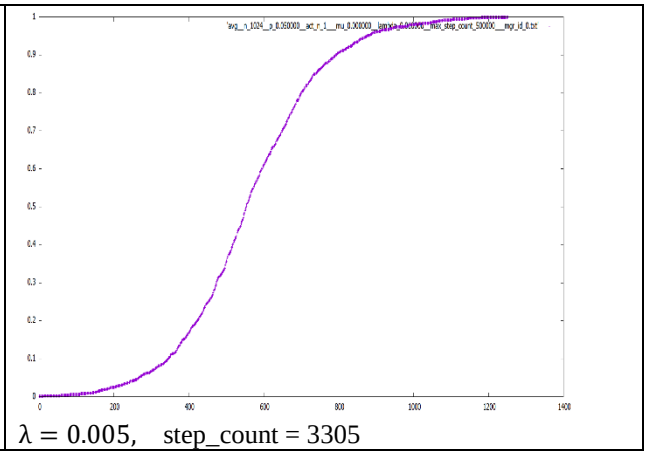
$\lambda = 0.1$, step_count = 210



$\lambda = 0.05$, step_count = 326



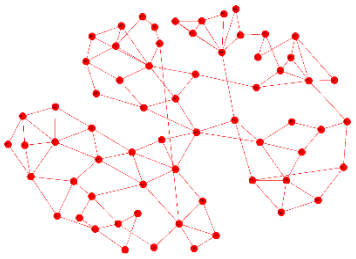
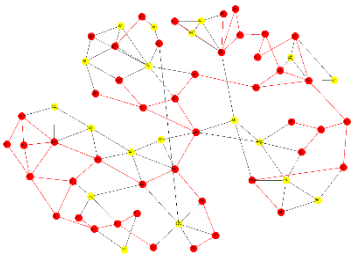
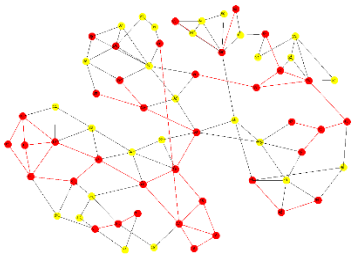
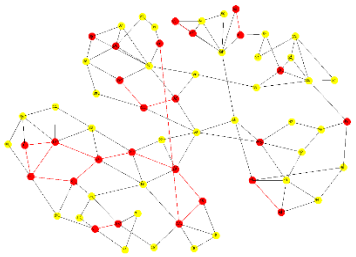
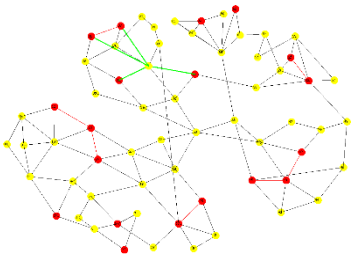
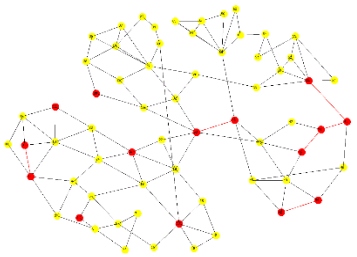
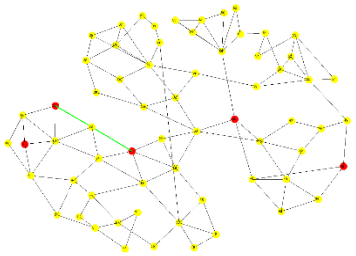
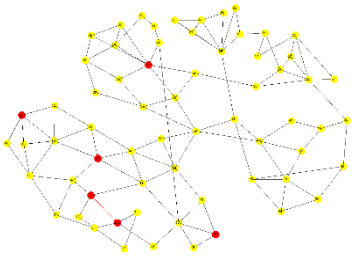
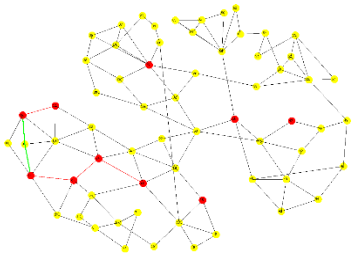
$\lambda = 0.01$, step_count = 1251



$\lambda = 0.005$, step_count = 3305

Приложение 2: Визуализация процесса распространения активности для сети с параметрами N=65, p = 0.05.

Таб. 25: Динамика процесса распространения активизации сети с параметрами: N=65, p=0.05, $\lambda = 0.314159265$, μ=1.

 *0*	 *20*	 *30*
 *50*	 *100*	 *500*
 *900*	 *1300*	 *1500*

ЗАКЛЮЧЕНИЕ

В работе были рассмотрены методы распространения активности в случайных сетях. Стоит повторить, что ни один из описанных алгоритмов не является универсальный и каждое из них имеет свои особенности, как преимущества так и недостатки. Вот почему описанные эксперименты позволяют выбрать алгоритм исходя из требований задачи.

Также поведения процесса распространения активности представляет большой интерес в иерархических и кластеризованных сетях, которым и будут посвящены будущие исследование.

Литература

1) Kriesel D. - A Brief Introduction to Neural Networks. 2005, 244
2) Moretti P. and Mun˜oz M. A. - Griffiths phases and the stretching of criticality in brain networks. 2013, 32
3) Newman M. - Networks: An Introduction. Oxford University Press Inc, 2010, 1042
4) Albert R. and Barabási A. L. - Statistical mechanics of complex networks. Department of Physics, University of Notre Dame, 2002, 54
5) Chilingarian I. and Klishin A. - Explaining the stellar initial mass function with the theory of spatial networks. The Astrophysical Journal 824, 1, 17, 2016
6) Chollet F. - Deep Learning with Python. Manning Publications Co. 2017, 368 pages
7) Fog A. - Optimizing software in C++. Technical University of Denmark. 2018, 168
8) Евин И.А. - Введение в теорию сложных сетей. //Компьютерные исследования и моделирование. 2010, Том 2, N2, с. 121-141
9) https://isocpp.org/files/papers/n3551.pdf 09/05/2018

-.