ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ
«ШАГ В БУДУЩЕЕ, МОСКВА»

1013

регистрационный номер

ИУ «Информатика и системы управления»

название факультета

ИУ7 «Программное обеспечение ЭВМ и информационные технологии»

название кафедры

Определение надёжности связи между двумя узлами компьютерной сети

название работы

Автор: Монахов Вадим Игоревич

фамилия, имя, отчество

ГБОУ города Москвы
"Измайловская школа № *1508*",
класс 10 «Б»

наименование учебного заведения, класс

Москва – 2021

Оглавление

1. Введение

В современном мире компьютерные сети, например, Интернет и телефонная сеть окружают нас и играют немаловажную роль в жизни человека. Компьютерная сеть — это система, обеспечивающая обмен данными между вычислительными устройствами — компьютерами, серверами, маршрутизаторами или другим оборудованием или программным обеспечением. При организации передачи данных между двумя узлами внутри компьютерной сети необходимо оценивать её надёжность, ведь важно понимать дойдёт ли информация до конечного пункта, например, получит ли банк своевременно запрос от банкомата о списании средств.

В данной проектной работе надёжность связи определяется как вероятность того, что данные, передаваемые из узла «A», успешно достигнут конечного узла «B». Эта работа посвящена методам определения этой величины.

Целью работы является сравнение способов определения надёжности связи между узлами в графе.

Таким образом, для решения поставленной цели мы можем сформулировать задачи работы:

1. Рассмотрение некоторых способов определения надёжности связи.
2. Сравнение экспериментального и аналитического методов решения.
3. Разработка программы для определения надёжности связи на произвольном графе.

2. Формат данных

Рассмотрим формат входных данных данной программы.

Компьютерная сеть вводится пользователем в виде графа, в котором узлы представляют компьютеры, датчики, ретрансляторы, маршрутизаторы и т.д., а грани представляют идеальный бесперебойный канал связи между двумя узлами сети.

Также пользователь осуществляет выбор начального и конечного узлов. Начальный узел – источник передаваемого сигнала. Конечный узел – пункт назначения передаваемого сигнала.

Каждому узлу сети сопоставлена его надежность. Надёжность узла – вероятность его исправной работы в момент передачи сигнала.

В качестве выходных данных программы пользователь получит:

- Вероятность того, что сигнал достигнет конечного узла, подсчитанный двумя способами;
- Рейтинг влияния узлов на надёжность связи.
- График зависимости относительной погрешности экспериментального метода от количества экспериментов.

3. Экспериментальный метод определения надёжности связи в сети.

Определение надёжности сети экспериментальным методом основано на проведении эксперимента большое количество раз. За счёт возможности изменять количество повторений эксперимента можно регулировать скорость и точность его работы.

Каждому узлу графа сопоставлена вероятность исправной работы соответствующего узла компьютерной сети. На основании этого значения программа выносит случайным образом решение об удалении узла из графа. Это имитирует поломку узла в сети. Затем программа проводит проверку существования хотя бы одного пути из начального узла в конечный узел. Если такой путь существует, то программа увеличивает значение счётчика успешных экспериментов. После проведения заданного количества экспериментов программа подсчитывает долю успешных экспериментов. Это число и будет примерно равняться надёжности связи между заданными узлами. На рисунке 1 изображена блок-схема данного алгоритма.

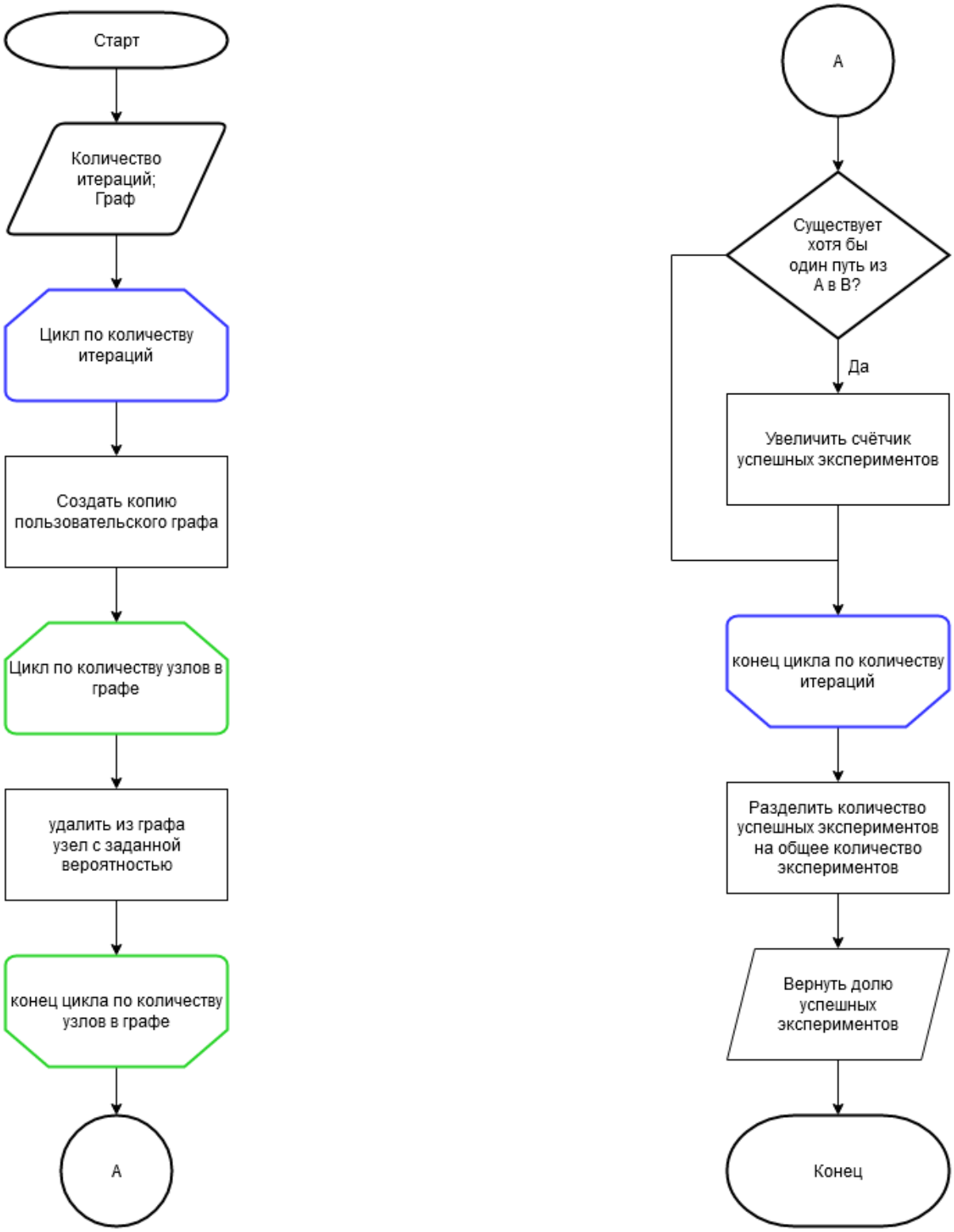

Рисунок 1. Блок-схема алгоритма экспериментального определения надёжности связи

4. Описания аналитического метода

Рассмотрим аналитический метод определения надежности связи.

Пусть имеется компьютерная сеть, представленная следующим графом (рис. 2):

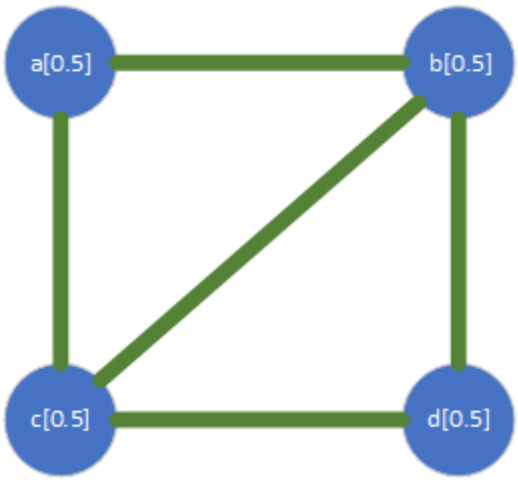

Рисунок 2. Пример графа компьютерной сети

Синие круги (вершины графа) соответствуют узлам компьютерной сети. Зеленые линии (грани графа) соответствуют каналам связи между ними. В центре каждого круга изображено название узла (a, b, c, d) и указана надежность данного узла в квадратных скобках. Для простоты предположим, что надежность каждого узла равна 0.5. С этой вероятностью узел будет корректно работать во время передачи сигнала.

Выберем узел *a* в качестве начального узла, а *d* в качестве конечного узла и попытаемся аналитическим путем определить вероятность прохождения сигнала. На время отстранимся от вероятностной природы поставленной задачи и будет оперировать событиями. Начнем с события «сигнал дошел». Обозначим его как событие Y. Нахождение вероятности наступления этого события является нашей задачей. Очевидно, что это составное событие, которое можно разложить на некоторые элементарные события. Попробуем определить эти составные события.

Начнем с рассмотрения узлов компьютерной сети. Корректная работа конкретного узла сети – это элементарное событие, имеющее только два исхода: «узел работал» и «узел сломался». Введем обозначения для этих событий. Пусть

событие A будет соответствовать работе узла a, событие B – работе узла b, событие C – работе узла c, а событие D – работе узла d.

Таким образом мы имеем одно составное событие Y и 4 элементарных события A, B, C, D. Найдем форму зависимости Y от названных элементарных событий. Нетрудно догадаться, что формулировка «сигнал дошел» тождественна формулировке «существовал хотя бы один путь из начального узла в конечный». Найдем все пути из узла a в узел b:

- a-b-d
- a-c-d
- a-b-c-d
- a-c-b-d

Запишем события, соответствующие корректной работе каждого пути:

- $P_1 = A \wedge B \wedge D$ (1)
- $P_2 = A \wedge C \wedge D$ (2)
- $P_3 = A \wedge B \wedge C \wedge D$ (3)
- $P_4 = A \wedge C \wedge B \wedge D$ (4)

Запишем логическое выражение для события Y:

$Y = P_1 \vee P_2 \vee P_3 \vee P_4$ (5)

Перепишем, подставив (1)-(4) в (5):

$Y = A \wedge B \wedge D \vee A \wedge C \wedge D \vee A \wedge C \wedge B \wedge D \vee A \wedge B \wedge C \wedge D$ (6)

Легко заметить, что операнды последней дизъюнкции эквивалентны:

$Y = A \wedge B \wedge D \vee A \wedge C \wedge D \vee A \wedge B \wedge C \wedge D$ (7)

Также можно применить закон поглощения для дизъюнкции:

$Y = A \wedge B \wedge D \vee A \wedge C \wedge D$ (8)

Здесь мы замечаем первую особенность: некоторые грани графа (a, соответственно, и каналы связи) можно удалить без какого-либо влияния на надежность связи между определенными узлами. В данном примере пути 3 и 4 зависят от путей 1 и 2. Такие пути далее будут именоваться сложными путями.

Сложный путь – это такой путь, который является надмножеством любого другого пути с точки зрения входящих в него узлов.

4.1 Теорема о сложении вероятностей

Продолжим рассмотрение формулы (8). Данная формула представляет собой логическое выражение, записанное в виде дизъюнктивной нормальной формы. Попробуем определить вероятность наступления события Y. Вспомним теорему о сложении вероятностей:

Вероятность появления одного из двух несовместных событий равна сумме вероятностей этих событий.

$$P(A \vee B) = P(A) \vee P(B)$$

Попробовав применить эту теорему к формуле (8), мы столкнемся с проблемой – события $A \wedge B \wedge D$ и $A \wedge C \wedge D$ являются совместными. Действительно, ничто не мешает тому, чтобы одновременно работали узлы a, b, c, и d. Эти события вполне могут произойти одновременно.

Рассмотрим обобщение сложения вероятностей для совместных событий:

Вероятность суммы совместных событий вычисляется по формуле:

$$P(A \vee B) = P(A) \vee P(B) - P(A \wedge B)$$

Расширим данную формулу на несколько событий. Здесь можно применить формулу включений-исключений – комбинаторную формулу, позволяющую определить мощность объединения конечного числа конечных множеств, которые в общем случае могут пересекаться друг с другом.

Формула включений-исключений:

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_i |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k| - \ldots + (-1)^{n-1} |A_1 \cap A_2 \cap \ldots \cap A_n|.$$

Вероятность суммы трех событий:

$$P(A \vee B \vee \mathrm{C}) = P(A) \vee P(B) \vee P(C) - P(A \wedge B) - P(A \wedge C) - P(B \wedge C) \vee$$
$$\vee P(A \wedge B \wedge C)$$

Видно, что количество слагаемых начинает очень быстро расти при увеличении количества событий. Определим зависимость количества слагаемых от количества событий в дизъюнкции:

$F(x) = \sum_{k=1}^{x} C_n^k$, где $C_n^k = \frac{n!}{k!(n-k)!}$

Становится понятно, что алгоритм, основанный на данной формуле, будет иметь сложность O(N!), что является неприемлемым. Поэтому следует рассмотреть другой подход.

4.2 Совершенная дизъюнктивная нормальная форма

Рассмотрим вариант нахождения вероятности события Y с помощью преобразования формулы (8) в такой вид, в котором исключено появление совместных событий. В таком случае можно будет применить формулу сложения вероятностей несовместных событий.

Одной из форм представления логических выражений является СДНФ (совершенная дизъюнктивная нормальная форма). СДНФ это частный случай ДНФ, удовлетворяющий следующим требованиям:

- в формуле нет одинаковых слагаемых;
- в каждом слагаемом нет повторяющихся переменных;
- каждое слагаемое содержит все переменные, от которых зависит логическое выражение.

СДНФ подходит для решения поставленной задачи. Из требований к СДНФ следует, что все составные события (слагаемые дизъюнкций) являются попарно несовместными, следовательно, к СДНФ можно применить формулу сложения вероятностей несовместных событий. Рассмотрим один из способов нахождения СДНФ для известного логического выражения.

4.3 Нахождение СДНФ по таблице истинности

Продолжаем рассматривать формулу (8). Для удобства, перепишем ее еще раз:

$Y = A \wedge B \wedge D \ \vee \ A \wedge C \wedge D$ (8)

Составим таблицу истинности данного выражения (таблица 1). Отметим только те строки, у которых присутствует единица в колонке Y. Каждая такая строка будет представлять слагаемое дизъюнкции. Каждое слагаемое состоит из конъюнкции всех переменных, от которых зависит выражение. Переменная в слагаемом входит в двух формах в зависимости от значения в ячейке на той же строке и в соответствующем столбце таблицы.

Запишем получившуюся СДНФ для выражения (8):

$$Y = (A \wedge \bar{B} \wedge \mathrm{C} \wedge D) \vee (A \wedge B \wedge \bar{\mathrm{C}} \wedge D) \vee (A \wedge B \wedge C \wedge D) \quad (9)$$

Таблица 1. Таблица истинности формулы (8)

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

И действительно, видно, что все слагаемые попарно не совместны. Следовательно, можно применить формулу сложения вероятностей для несовместных событий:

$$P(Y) = P(A \wedge \bar{B} \wedge \mathrm{C} \wedge D) \vee P(A \wedge B \wedge \bar{\mathrm{C}} \wedge D) \vee P(A \wedge B \wedge C \wedge D)$$

Распишем каждое слагаемое в соответствии с теоремой об умножении вероятностей. События A, B, C, D являются элементарными, следовательно, независимыми, поэтому:

$P(Y) =$

$P(A) * P(\bar{B}) * P(\mathrm{C}) * P(D) +$

$P(A) * P(B) * P(\bar{\mathrm{C}}) * P(D) +$

$P(A) * P(B) * P(C) * P(D)$

Подставим значения надежности соответствующих узлов. Для событий, входящих с инверсией, будем использовать вероятность отказа узла, которая вычисляется по формуле $P(\bar{X}) = 1 - P(X)$

$$P(Y) = 0.5^4 + 0.5^4 + 0.5^4 = 0.1875$$

Таким образом, вероятность того, что сигнал из узла a достигнет узла b составляет 0.1875.

Рассмотрим недостаток данного подхода. Он заключается в составлении таблицы истинности. Количество строк в таблице истинности равняется 2^N, где N – количество узлов в исходной сети. Следовательно, алгоритму потребуется 2^N раз вычислить логическое выражение. Алгоритм обладает сложностью $O(2^N)$, что достаточно много. В связи с этим рассмотрим другой способ составления СДНФ.

4.5 Вывод СДНФ из ДНФ

Рассмотрим другой граф (рис. 3).

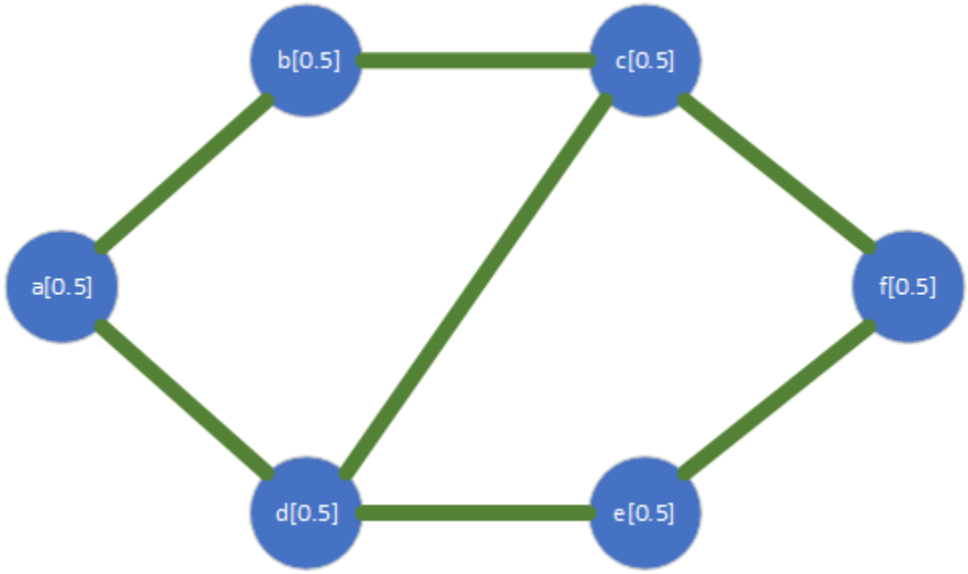

Рисунок 3. Граф компьютерной сети

Выберем a и d в качестве начального и конечного узлов. Аналогично предыдущему примеру найдем все пути из a в d:

1. a-b-c-f
2. a-d-e-f
3. a-d-c-f
4. a-b-c-d-e-f

Сразу же определяем, что путь 4 является сложным, т.к. является надмножеством маршрута a-b-c-f. Удаляем его из рассмотрения.

Запишем ДНФ для события Y. Для простоты здесь и далее будем опускать символ оператора конъюнкции:

$$Y = ABCF \vee ADEF \vee ADCF \quad (10)$$

Найдем СДНФ любым из уже рассмотренных способов и попытаемся найти какие-либо закономерности:

$$Y = ADEF\bar{B}\bar{C} \vee AB\bar{C}DEF \vee A\bar{B}CDEF \vee ACDF\bar{B}\bar{E} \vee ABCD\bar{E}F \vee ABCF\bar{D}\bar{E} \vee AB\bar{C}\bar{D}EF \vee ABCDEF \quad (11)$$

Следует обратить внимание на то, что любое слагаемое в СДНФ можно получить из слагаемого в ДНФ путем добавления недостающих множителей и добавления нового слагаемого, которое компенсирует изменения. Например:

$$ABCF = ABCF\overline{D}\overline{E} \vee ABCF\left(\overline{\overline{D}\overline{E}}\right)$$
$$= ABCF\overline{D}\overline{E} \vee ABCF\left(\overline{\overline{D}} \vee \overline{\overline{E}}\right)$$
$$= ABCF\overline{D}\overline{E} \vee ABCF(D \vee E)$$
$$= ABCF\overline{D}\overline{E} \vee ABCFD \vee ABCFE$$
$$(11)$$

Также преобразуем новые слагаемые:

$$ABCFD = ABCFD\overline{E} \vee ABCFDE \ (12)$$
$$ABCFE = ABCF\overline{D}E \vee ABCFDE \ (13)$$

Подставим (12) и (13) в (11) и упростим:

$$ABCF\overline{D}\overline{E} \vee ABCFD \vee ABCFE$$
$$= ABCF\overline{D}\overline{E} \vee ABCFD\overline{E} \vee ABCFDE \vee ABCF\overline{D}E \vee ABCFDE$$
$$= ABCF\overline{D}\overline{E} \vee ABCFD\overline{E} \vee ABCF\overline{D}E \vee ABCFDE$$

Обратим внимание, что, взяв одно слагаемое из ДНФ и применив несложные преобразования, мы получили фрагмент СДНФ. Проведя аналогичные преобразования для других слагаемых из ДНФ и упростив результат, мы получим искомую СДНФ. Можно заметить, что данный алгоритм легко автоматизировать. Необходимо для каждого слагаемого ДНФ:

- найти узлы, которые в нем отсутствуют;
- найти комбинаций отсутствующих узлов, где каждый узел может быть с инверсией или без нее, (в примере выше для слагаемого ABCF это комбинации $\overline{D}\overline{E}, \overline{D}E, D\overline{E}, DE$);
- добавить к каждой комбинации исходное слагаемое в виде операнда конъюнкции и дописать к выражению (в примере выше это слагаемые $ABCF\overline{D}\overline{E}, ABCFD\overline{E}, ABCF\overline{D}E, ABCFDE$);

Для некоторых слагаемых комбинации будут повторяться. Это необходимо учитывать и не добавлять в СДНФ слагаемые, которые в нем уже присутствуют, чтобы избежать трудоемкого с алгоритмической точки зрения упрощения итогового выражения.

Данную закономерность можно использовать для более эффективного построения СДНФ. На рисунке 4. приведена блок-схема разработанного алгоритма.

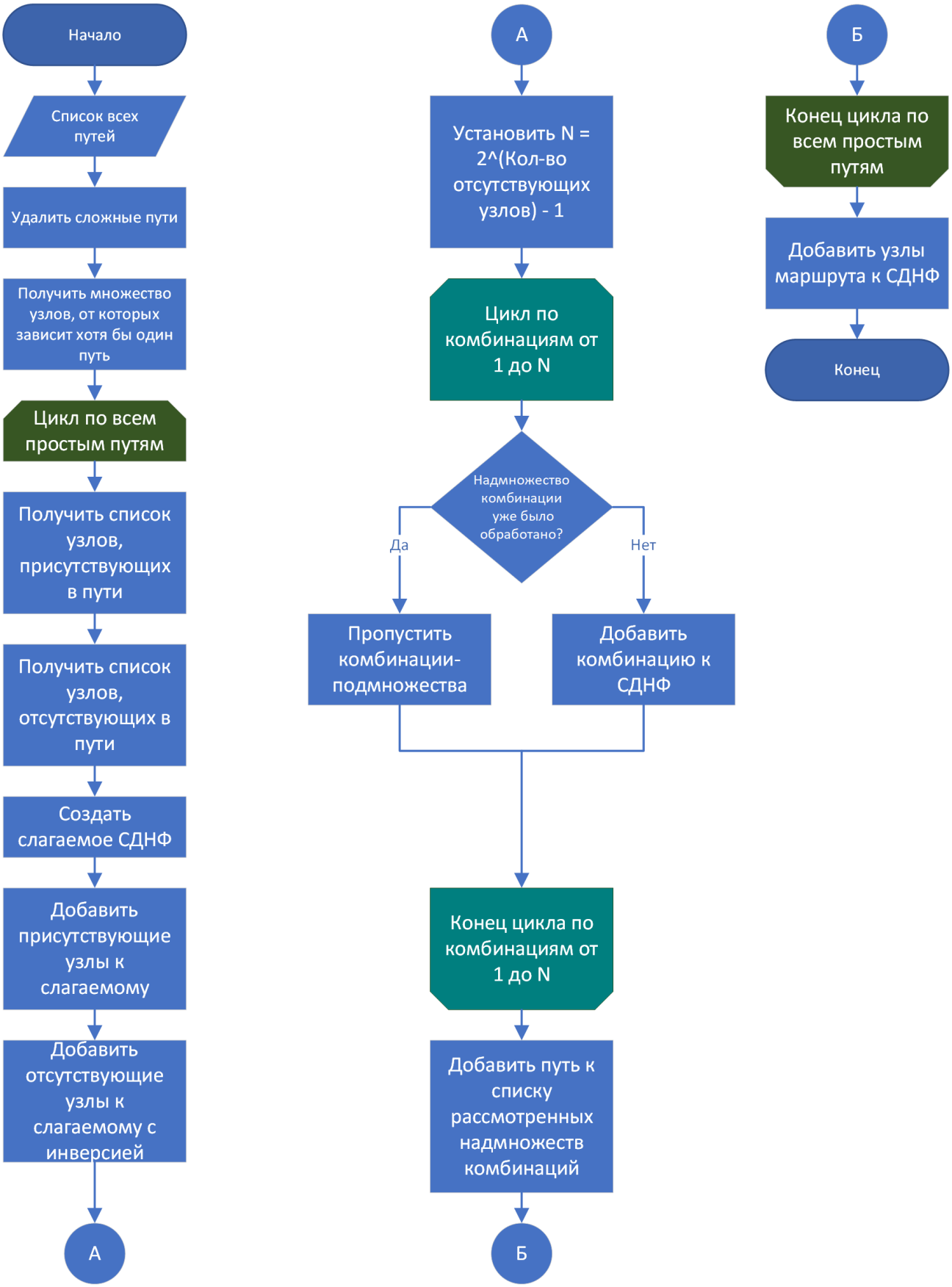

Рисунок 4. Блок-схема алгоритма построения СДНФ

В реализации данного алгоритма используются битовые операции вместо массивов, что позволяет значительно ускорить работу. В частности, слагаемые СДНФ представляются в виде целого числа в двоичной форме. Каждый бит соответствует узлу компьютерной сети и указывает, входит ли узел в данном слагаем в прямой или инверсной форме. Битовые операции также ускоряют вычисление количества комбинаций (каждая комбинация также представляется целым числом).

Сохранение списка уже обработанных слагаемых позволяет пропускать часть итераций внутреннего цикла, что также значительно ускоряет работу алгоритма.

Предложенный алгоритм обладает сложностью, меньшей, чем экспоненциальная, что делает его более эффективным по времени, чем рассмотренные выше. Также данный алгоритм зависит от количества путей, а не от количества узлов, что делает его еще более эффективным на некоторых видах графов.

5. Описание вспомогательных алгоритмов

5.1 Поиск любого пути из начального узла в конечный узел.

Поиск любого пути из начального узла в конечный узел, осуществляется благодаря поиску в ширину - одному из методов обхода графа.

5.2 Работа алгоритма

Поиск в ширину работает путём последовательного просмотра отдельных уровней графа, начиная с узла-источника **u**.

Рассмотрим все рёбра (**u**, **v**), выходящие из узла **u**. Если очередной узел **v** является целевым узлом, то поиск завершается; в противном случае узел **v** добавляется в очередь. После того, как будут проверены все рёбра, выходящие из узла **u**, из очереди извлекается следующий узел **u**, и процесс повторяется.

Неформальное описание алгоритма поиска в ширину:

1) Поместить узел, с которого начинается поиск, в изначально пустую очередь.
2) Извлечь из начала очереди узел и пометить его как развёрнутый.
3) Если текущий узел является целевым узлом, то завершить поиск с результатом «успех».
4) В противном случае, в конец очереди добавляются все преемники текущего узла, которые ещё не развёрнуты и не находятся в очереди.
5) Если очередь пуста, то все вершины связного графа были просмотрены, следовательно, целевой узел недостижим из начального; завершить поиск с результатом «неудача».
6) Вернуться к п. 2.

5.3 Поиск всех путей в графе из начального узла в конечный узел.

Поиск всех путей в графе из начального узла в конечный узел, осуществляется благодаря алгоритму поиска в глубину.

На рисунке 5 представлена блок-схема данного алгоритма.

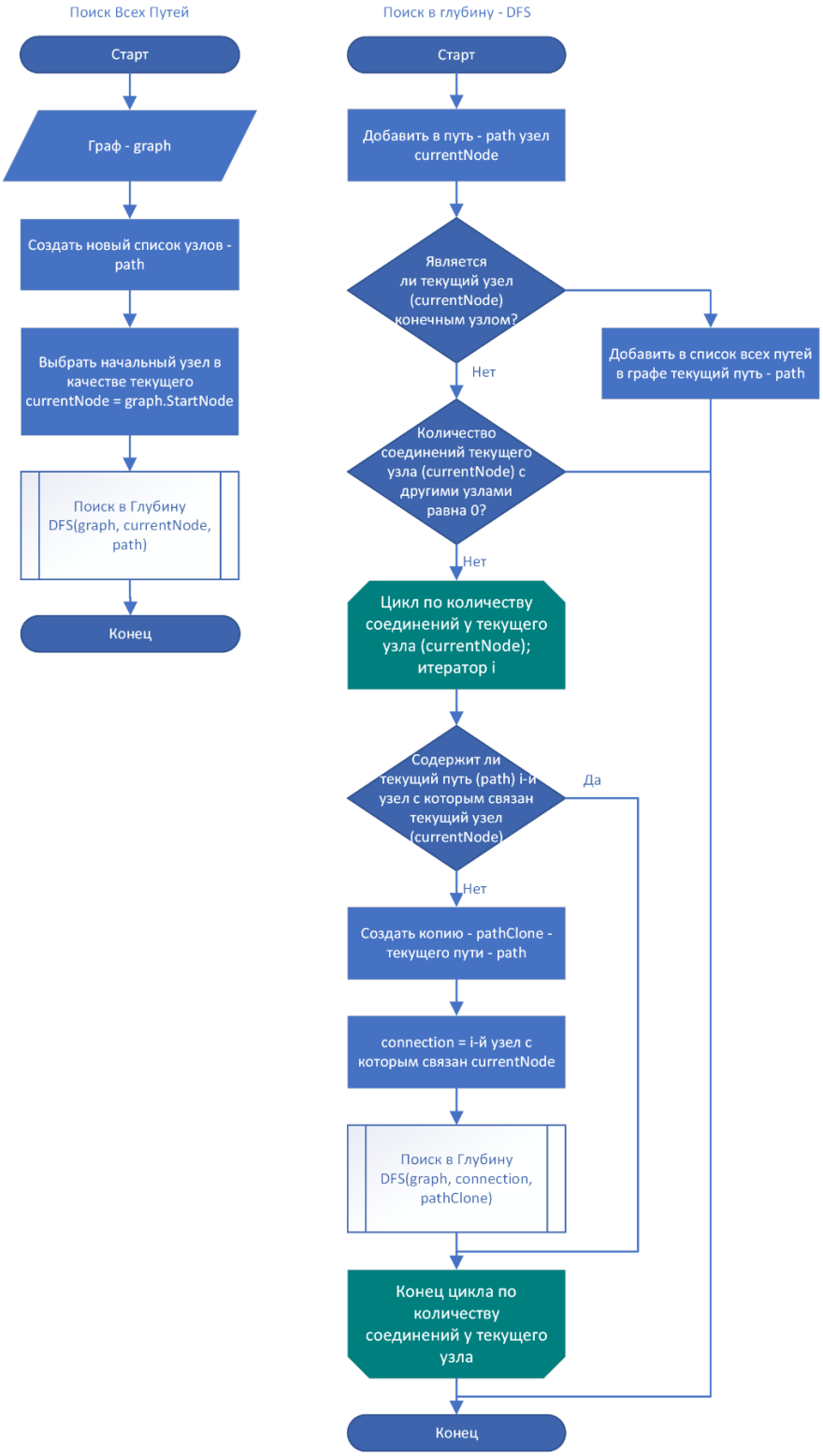

Рисунок 5. Блок-схема алгоритма нахождения всех путей

6. Описание структуры проекта

6.1 Диаграмма классов

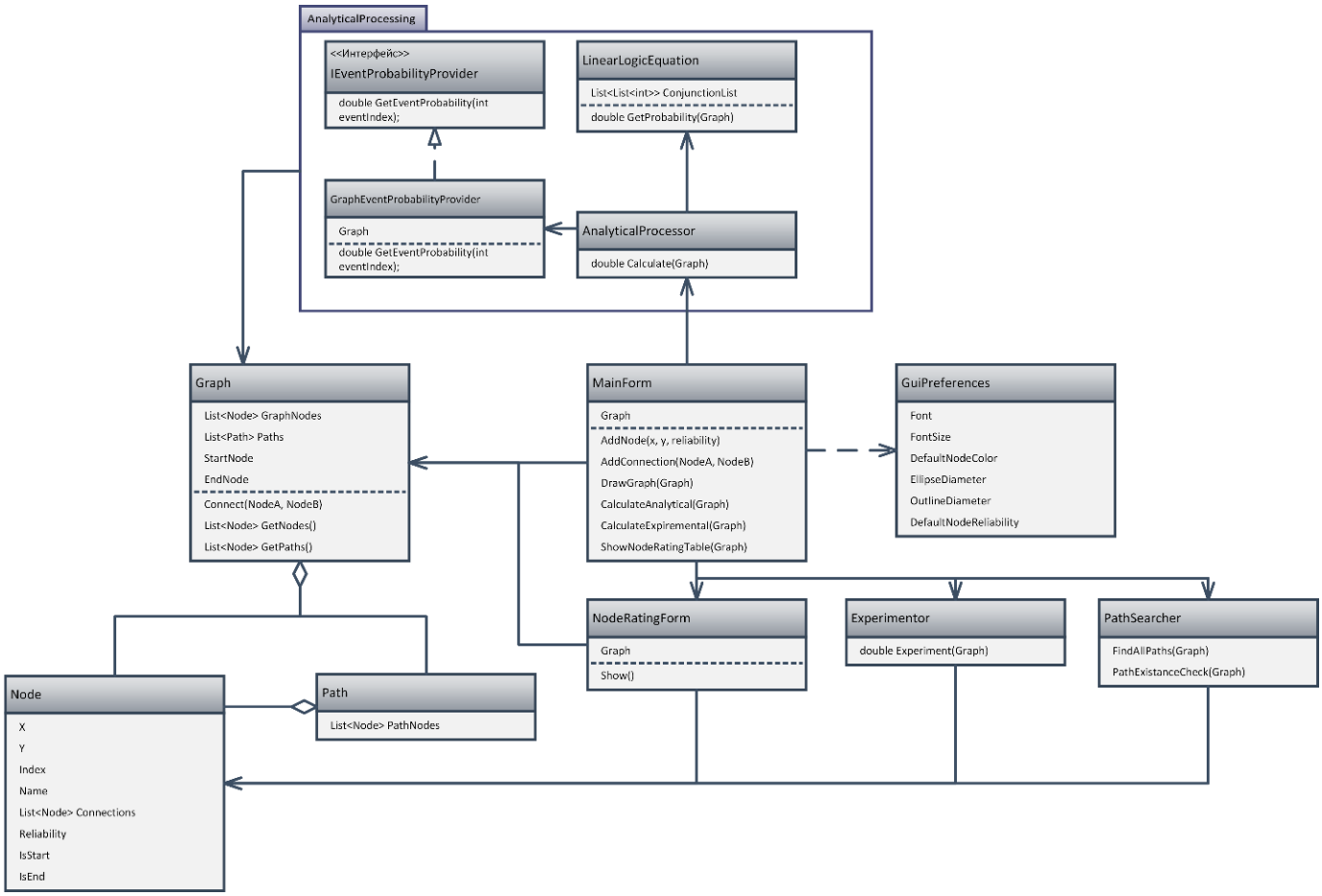

Рисунок 6. Диаграмма классов

6.2 Краткое описание классов

- **MainForm** – класс представляющий собой основное окно взаимодействия с пользователем. В данном окне происходит построение модели компьютерной сети, установка начального и конечных узлов и т.п.
- **NodeRatingForm** – окно взаимодействия с пользователем, которое выводит пользователю таблицу рейтинга влияния узлов.
- **GuiPreferences** – класс для хранения настроек пользовательского интерфейса, например, размер узлов на экране пользователя.

- **Graph** – набор узлов и связей между ними, моделирующий граф. Хранит начальный и конечный узлы, а также список путей из начального узла в конечный.
- **Node** – узел в графе. Хранит надёжность соответствующего узла компьютерной сети, а также список соединённых с ним других узлов.
- **Path** – последовательность узлов (путь).
- **Experimentor** – класс, занимающийся вычислением надёжности связи между начальным и конечным узлами экспериментальным методом.
- **PathSearcher** – класс выполняющий поиск всех путей из начального узла в конечный, а также проверку существования хотя бы одного любого пути из начального узла в конечный.
- **AnalyticalProcessor** – класс, реализующий аналитический метод определения надёжности связи между начальным и конечным узлами.
- **LinearLogicEquation** – класс, представляющий логическое выражение.
- **GraphEventProbabilityProvider** – класс, позволяющий определить надёжность узла по его индексу из логического выражения.
- **IEventProbabilityProvider** – интерфейс, представляющий собой конвертер индекса события в его вероятность.

7. Пользовательский интерфейс

Пользовательский интерфейс реализован при помощи библиотеки windows forms. Программа имеет одно основное окно (рисунок 7) взаимодействия с пользователем. В правой части основного окна расположена строка ввода и кнопки управления:

- «сохранить данные из строки ввода»
- «установить начальный узел»
- «установить конечный узел»
- «вычислить надёжность связи экспериментальным методом»
- «вычислить надёжность связи аналитическим методом»
- «получить рейтинг влияния надёжности узлов»
- «получить график относительной погрешности экспериментального метода»

Большую часть основного окна занимает рабочее пространство для построения модели компьютерной сети, представленной графом.

Создание нового узла в произвольной точке рабочего пространства осуществляется двойным щелчком мыши.

Чтобы создать соединение между двумя узлами необходимо щелчком левой кнопки мыши выбрать любой узел (узел станет светло-синим), а затем правой кнопкой мыши нажать на другой узел, с которым пользователь хочет создать связь.

Для удаления связи между связанными узлами необходимо аналогично созданию связи выбрать первый узел нажатием левой кнопки мыши, а затем правой кнопкой мыши нажать на связанный узел.

Выбор начального и конечного узлов осуществляется нажатием левой кнопкой мыши на желаемый узел и нажатием соответствующей кнопки на панели в правой стороне окна приложения.

Для вычисления надёжности связи между начальным и конечным узлами необходимо нажать на соответствующие кнопки, на панели в правой стороне окна приложения. Результат вычисления появится во всплывающем окне.

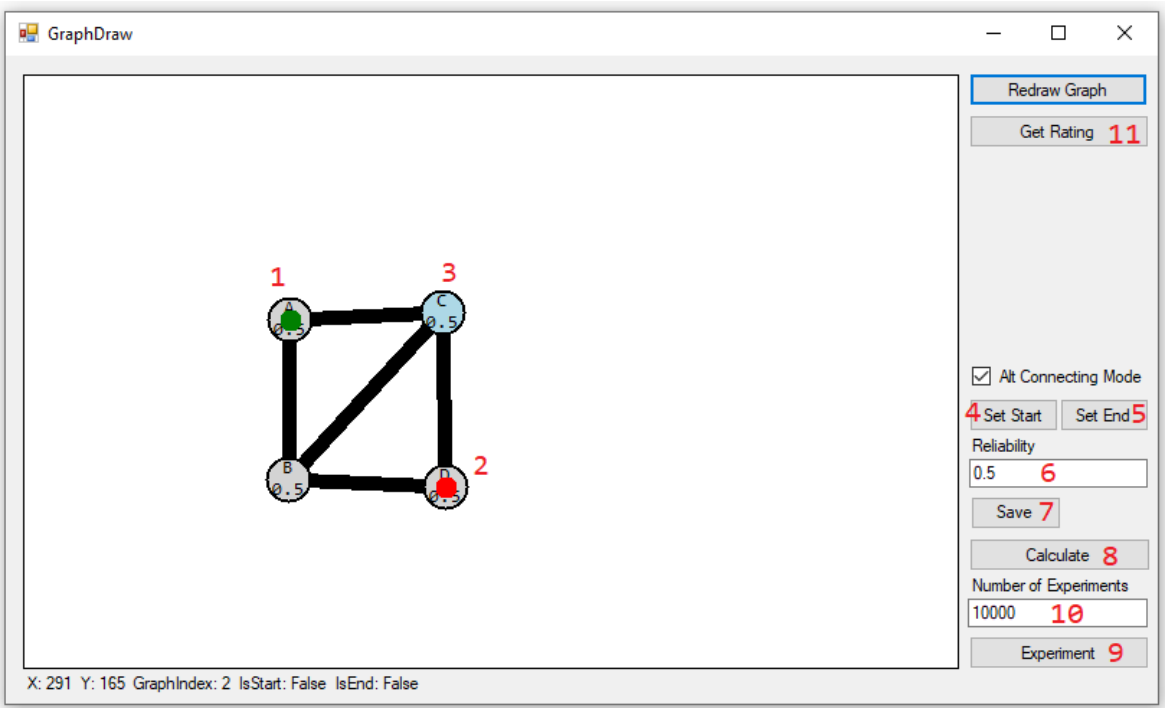

Рисунок 7. 1 - Начальный узел; 2 - Конечный узел; 3 - Текущий узел; 4 - Установить начальный узел; 5 - Установить конечный узел; 6 - Поле ввода надёжности текущего узла; 7 - Сохранить изменения в поле для ввода надёжности текущего узла; 8 – Вычислить надёжность связи аналитическим методом; 9 - Вычислить надёжность связи экспериментальным методом; 10 – Количество экпериментов; 11 – Получить рейтинг влияния узлов на надёжность связи; 12 – получить график относительной погрешности экспериментального метода;

Нажав на кнопку получения рейтинга узлов, пользователь может получить сводку о влиянии каждого из узлов сети на надёжность связи. После нажатия на соответствующую кнопку появится новое окно с таблицей. Пример данной таблицы представлен на рисунке 8. В первой колонке буквами латинского алфавита записаны названия узлов. Во второй колонке указано число, обозначающее сколько раз соответствующий узел, был задействован во всех маршрутах из начального узла в конечный. В третьей колонке находятся значения надёжности узлов. В четвёртой колонке находятся значения «рейтинга важности» каждого из узлов. Данное число упрощённо показывает степень негативного влияния ненадёжных узлов на сеть. Число в этой колонке может варьироваться от 0 до 1. Рейтинг равный «0» получают те узлы, которые не оказывают негативного влияния на надёжность связи между

двумя выбранными узлами. Рейтинг больший нуля получают узлы препятствующие связи между начальным и конечным узлами. Чем больше это число, тем сильнее данный узел влияет на надёжность связи. При выводе данной таблицы, программа автоматически сортирует строки по значениям этого столбца. Таким образом пользователь может сделать вывод о наличии узких мест в сети и определить какие узлы требуют повышенного влияния. Например, пользователь может сделать вывод о необходимости повышения надёжности узлов с наибольшим рейтингом.

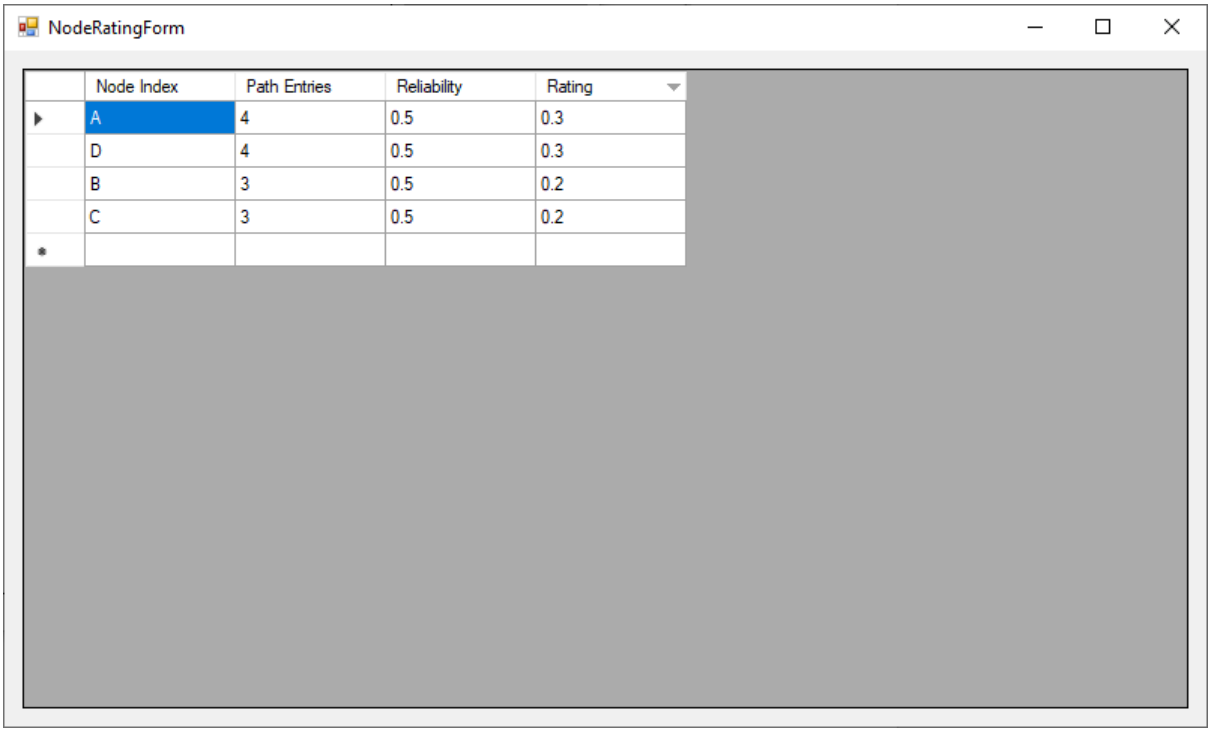

Рисунок 8. Пример таблицы рейтинга влияния узлов

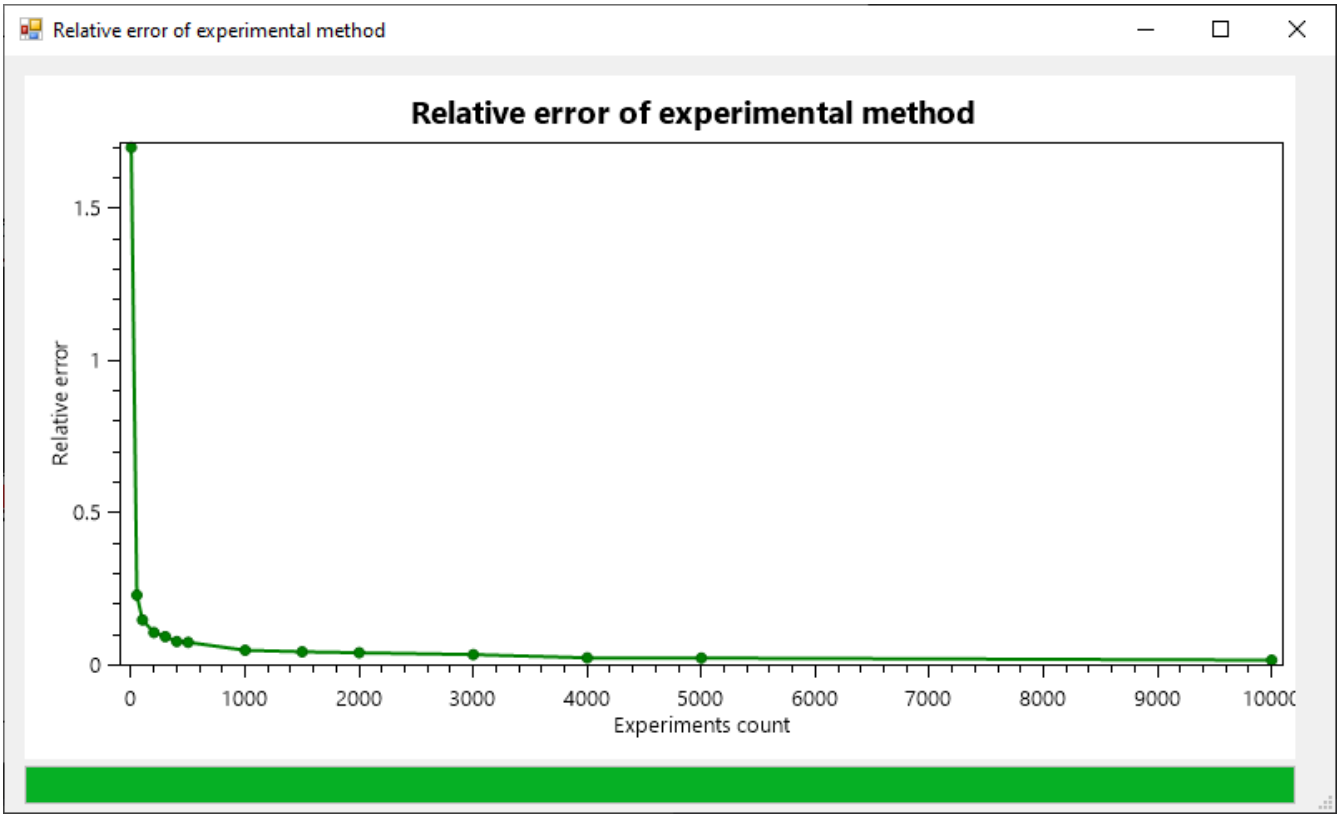

Рисунок 9. График относительной ошибки экпериментального метода

8. Сравнение методов

Аналитический метод является точным, но при этом время его работы субэкспоненциально зависит от количества путей в построенной пользователем сети. Экспериментальный метод превосходит аналитический метод по скорости на больших графах за счёт своей линейной сложности, а также возможности регулировать количество экспериментов, а следовательно, и скорость его работы, но при этом жертвуя точностью результата. На рисунках 9 и 10 представлен пример модели компьютерной сети в виде сетки 5 на 5. Аналитический метод обрабатывает эту модель за 6,5 – 6,7 секунд. А экспериментальный метод с количеством экспериментов в 100 000 выдаёт результат за ~3,5 секунды. При этом погрешность экспериментального метода составляет около 0,07%.

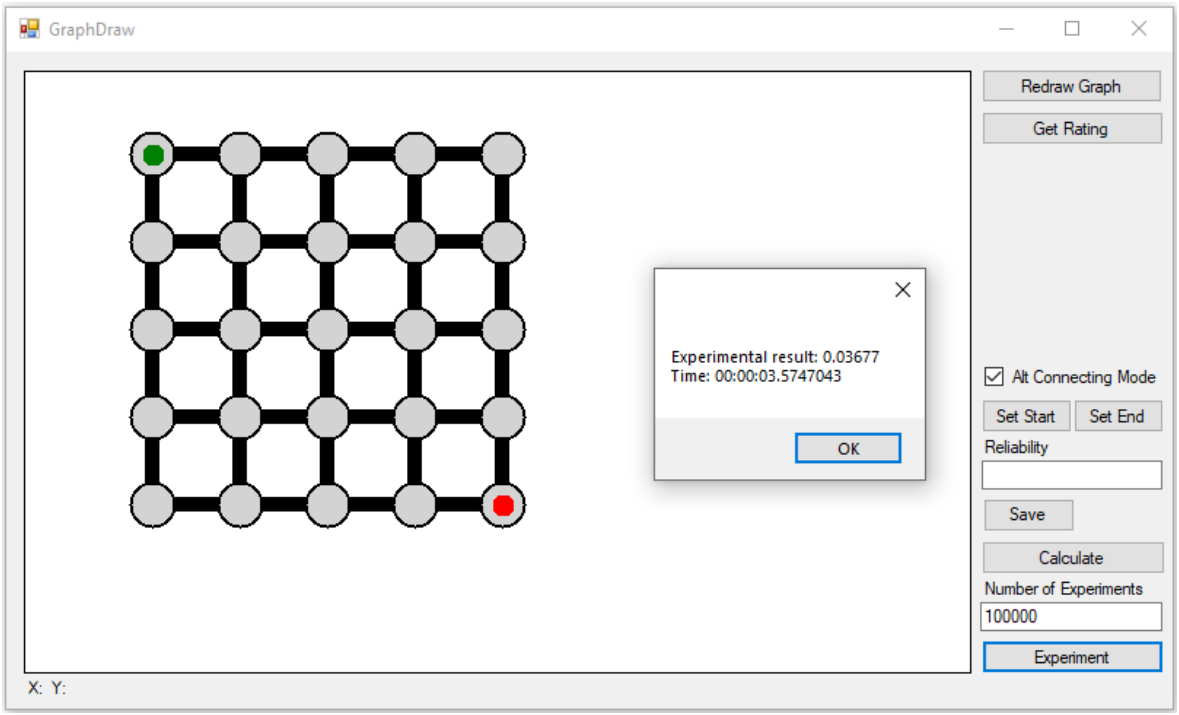

Рисунок 10. Результат экспериментального метода

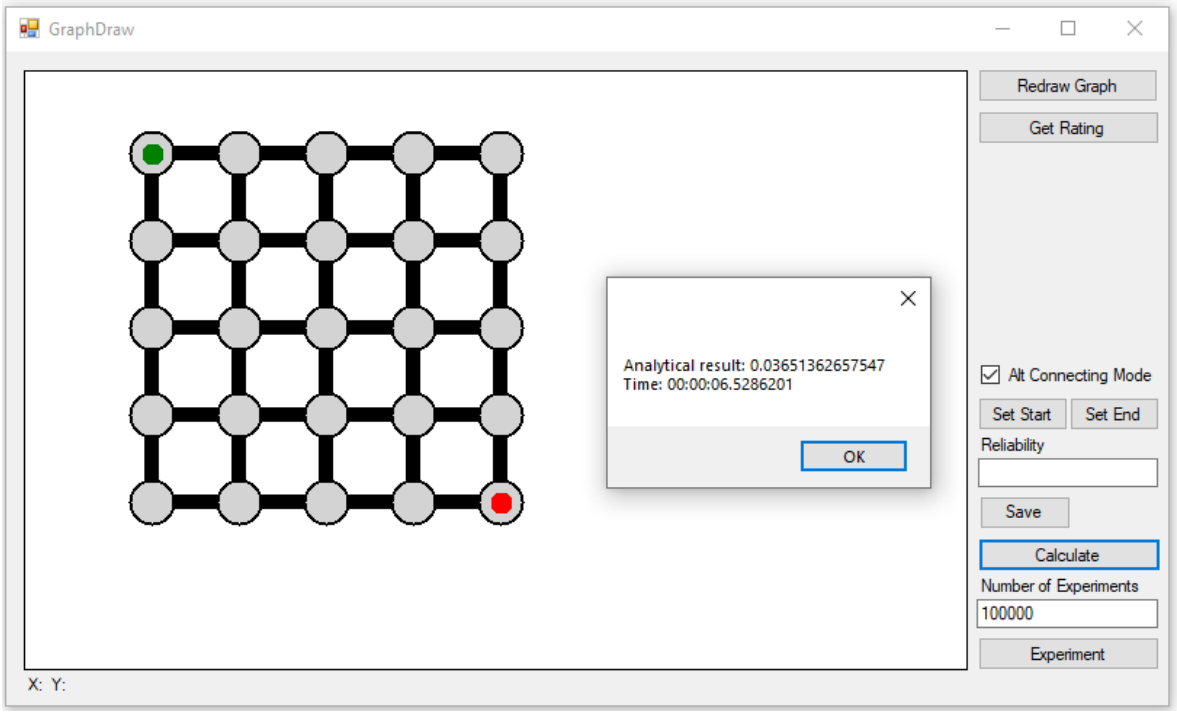

Рисунок 11. Результат аналитического метода

Теперь рассмотрим ту же модель с двумя дополнительными узлами. Как показано на рисунке 11, время работы аналитического алгоритма возросло в несколько раз. Это связано с тем, что добавление новых узлов вызвало многократное увеличения количества путей. В то же время экспериментальный алгоритм (рисунок 12) решил задачу примерно так же быстро, как и с предыдущей моделью. Дело в том, что экспериментальный метод линейно зависит от количества узлов в графе. Это делает его более пригодным для применения на практике. Также преимуществом данного алгоритма можно назвать возможность варьировать скорость и точность, подстраивая его под текущие нужды пользователя и возможности его вычислительной машины.

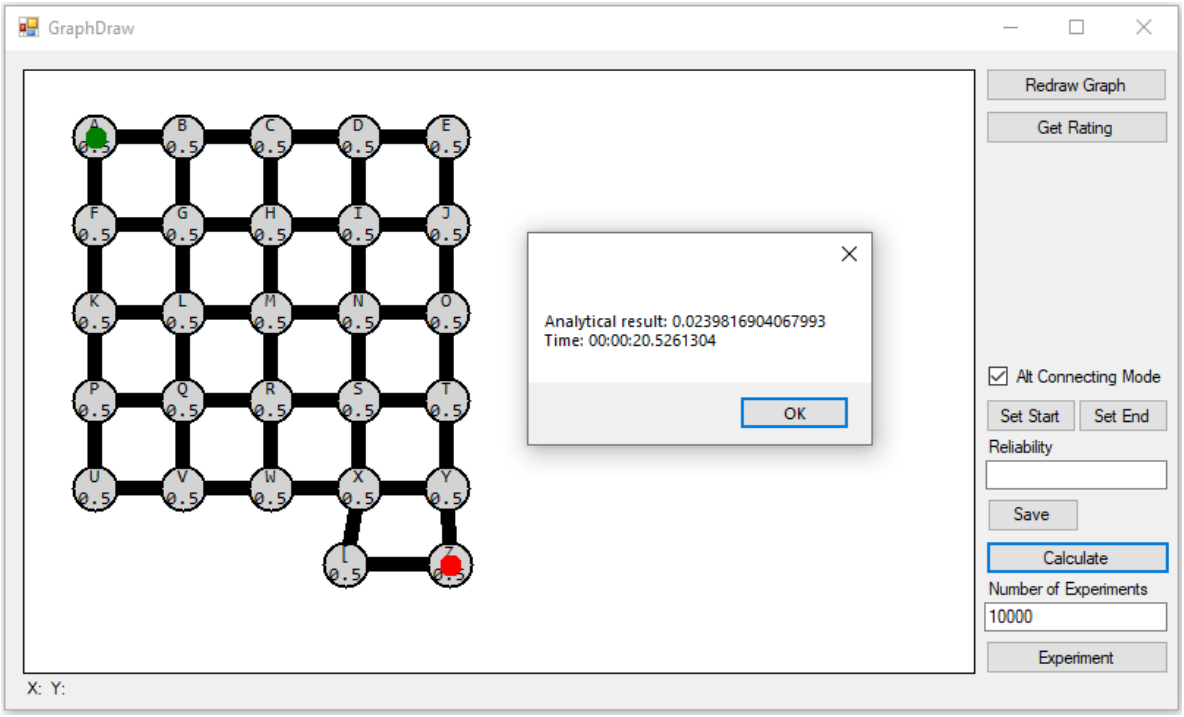

Рисунок 12. Пример работы аналитического метода

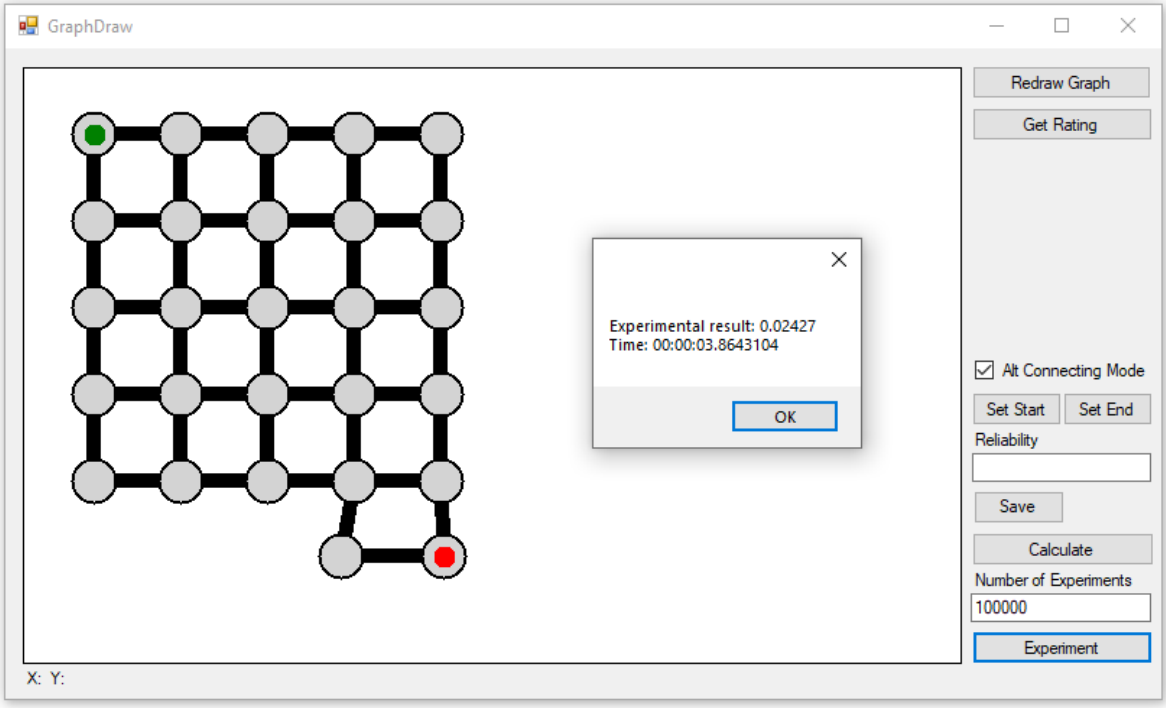

Рисунок 13. Пример работы экспериментального метода

9. Вывод

В ходе работы были достигнуты все поставленные цели. Были рассмотрены несколько способов определения надёжности связи. Осуществлено сравнение аналитического и экспериментального способов определения надёжности связи.

Было показано, что экспериментальный способ обладает большей эффективностью по времени на больших графах, а его точность регулируется количеством итераций. Аналитический же способ всегда выдаёт точный результат, но в силу своей субэкспоненциальной сложности, его становится трудно применять на достаточно больших графах.

Конечным результатом работы стала разработанная программа для определения надёжности связи между двумя узлами на произвольном графе с применением рассмотренных алгоритмов. Разработанная программа наглядно демонстрирует различия между рассмотренными подходами, позволяет анализировать предложенные алгоритмы с точки зрения их применения к компьютерным сетям различной конфигурации. Также программа имеет прикладное значение в виде поиска «узких мест» в компьютерной сети пользователя.