Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики

Работа допущена к защите
Руководитель ОП
_____ К.Н. Козлов
«___»_____20__г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

АЛГОРИТМИЧЕСКИЙ ПОДХОД К ИССЛЕДОВАНИЮ МИНИМАЛЬНЫХ И МИНИМАЛЬНЫХ ПО СТЯГИВАЮ *K*-СВЯЗНЫХ ГРАФОВ

по направлению подготовки 01.03.02 «Прикладная математика и информатика»

Направленность 01.03.02_02 «Системное программирование»

Выполнил
студент гр.3630102/70201 Д.А. Зубкова

Руководитель
к.ф.-м.н, доцент. ВШПМиВФ А.В. Пастор

Консультант
старший преподаватель ВШПМиВФ В.С. Чуканов

Санкт-Петербург
2021

РЕФЕРАТ

На 38 с., 13 рисунков, 5 таблиц.

K-СВЯЗНОСТЬ, МИНИМАЛЬНОСТЬ, МИНИМАЛЬНОСТЬ ПО СТЯГИВАНИЮ, ГРАФЫ.

Тема выпускной квалификационной работы: «Алгоритмический подход к исследованию минимальных и минимальных по стягиванию k-связных графов».

Работа посвящена разработке алгоритма проверки графов на свойства минимальности, минимальности по стягиванию и k-связности.

Цель работы — поиск минимальных и минимальных по стягиванию k-связных графов. Настоящая выпускная квалификационная работа содержит обзор существующих результатов исследований рассматриваемых графов, методов проверки на k-связность и вспомогательных методов. Исходя из рассмотренных данных, предложен алгоритм проверки входящих графов на свойства минимальности, минимальности по стягиванию и k-связности, а также поиск графов, обладающих такими свойствами внутри заданного на вход графа.

Результатом работы является программа, реализующая алгоритм проверки графов на свойства k-связности, минимальности и минимальности по стягиванию, алгоритм поиска графов, обладающих предыдущими свойствами.

THE ABSTRACT

На 38 pages, 13 pictures, 5 tables.

K-CONNECTION, MINIMUM, CONTRACTION MINIMUM, GRAPHS.

The subject of the graduate qualification work is «Algorithmic approach for minimal and minimal contraction k-connected graphs research».

The work is devoted to the development of an algorithm for checking graphs for the properties of minimality, minimality in contraction and k-connectivity.

The work is devoted to the development of an algorithm for checking graphs for the properties of minimality, minimality in contraction and k-connectivity. The purpose of this work is to find minimal and minimal contraction k-connected graphs. This final qualifying work contains an overview of existing research results of the considered graphs, k-connectivity verification methods and auxiliary methods. Based

on considered data, an algorithm is proposed for checking incoming graphs for properties of minimality, contraction minimality and k-connectivity, as well as for searching for graphs with such properties inside the graph given at the input.

The result of the work is a program that implements an algorithm for checking graphs for the properties of k-connectivity, minimality and contraction minimality, an algorithm for finding graphs with the previous properties.

СОДЕРЖАНИЕ

ВВЕДНИЕ

На сегодняшний день не существует полностью безотказных сетей, но чем выше бесперебойность их работы при наличии сбоев и отказов, тем эффективнее система поддерживает рабочее состояние. Отказоустойчивость, как важное свойство технической системы, требуется в разных сферах жизнедеятельности, но топология таких отказоустойчивых сетей при минимальных затратах очень сложна. Кроме того, при топологическом проектировании adhoc-сетей имеются ограниченные ресурсы энергии, памяти и вычислительной мощности узла. Также построение такой сети должно соответствовать достаточной вычислительной эффективности и надежности.

Из соображений надежности и безопасности от других сетей нужно предоставлять более одного отдельного пути для каждой пары узлов, что приводит к минимальной степени связности между вершинами. Таким образом, надежность является одним из наиболее важных понятий в топологическом построении сети, которую можно описать связностью узлов. А связность, в свою очередь, является ключевой детерминированной мерой отказоустойчивости в теории графов.

Когда одна вершина или компонента сети выходит из строя, то её функции должны быть переданы остальным рабочим звеньям. Отказоустойчивая сеть должна продолжать работу в случае сбоя какой-либо компоненты. Существует множество интерпретаций понятия «отказоустойчивость». Например, Aviziens определила отказоустойчивость в компьютерной сети связи, как «способность системы правильно выполнять заданные алгоритмы независимо от аппаратного сбоя и ошибок программы».[1] Однако, со стороны теории графов понятие отказоустойчивости сводится к возможности связности узлов сети, как это следует из определения надежности, то есть минимальному количеству вершин, которые при удалении уничтожают все существующие пути между какой-нибудь парой узлов. При этом желательна наибольшая возможная связность, так как это отвечает не только за максимальную отказоустойчивость, но и за наибольшее количество связанных непересекающихся путей между любыми двумя разными

узлами. Следует учесть, что вершинная связность ограничена сверху наименьшей степенью вершины. Следовательно, для построения отказоустойчивой сети достаточно иметь k-связный граф в виде его топологической структуры.

В данной работе осуществляется программный поиск минимальных и минимальных по стягиванию k-связных графов. Актуальность данной работы обосновывается тем, что с помощью данного алгоритма и его реализации можно проверить является ли входная сеть, имеющая структуру графа, отказоустойчивой. Таким образом, в процессе выполнения алгоритма производится поиск минимальных и минимальных по стягиванию k-связных графов, где в случае успешного нахождения таких графов можно сделать вывод, что сеть, поступившая на вход программы, не является отказоустойчивой.

ГЛАВА 1. ОБЗОР ЛИТЕРАТУРЫ

1.1. Основные определения

Граф $G(V,E)$ – это совокупность непустого множества вершин $V(G)$ и множества ребер $E(G)$. Пусть $G(V,E)$ будет конечным неориентированным графом без петель и кратных ребер. Иными словами, *граф* – это определенное количество узлов (вершин), которые могут быть соединены между собой ребрами или быть изолированными (не иметь инцидентных ребер). Если невозможно пойти по ребру и прийти в ту же вершину, то это значит, что граф не имеет петель. Если вершины в графе могут быть соединены только одним ребром или не соединены вообще, то говорят, что граф не имеет кратных ребер.

Связанный граф – это граф, в котором между любыми двумя его различными вершинами существует путь. *Путем* считается конечная последовательность ребер, в которой между двумя соседними ребрами существует одна общая вершина. В данном случае путь является простым и не может пройти через одну вершину более одного раза.

Рассмотрим понятие вершинной связности графа $\kappa(G)$. Для начала нужно дать определение смежным понятиям:

— *степенью* $d(v)$ вершины v называется количество инцидентных этой вершине ребер, то есть количество ребер, выходящих из данного узла, если граф неориентированный. Для ориентированного графа существуют соответствующие понятия: *степень входа вершины* – число входящих ребер и *степень выхода вершины* – число выходящих ребер;

— подграф, индуцированный множеством вершин A, будет обозначатся, как $G(A)$. Если $A \subset V(G)$, то $G - A$ – это граф, полученный из G, удалением вершин множества A и инцидентных им ребер.

Теперь, благодаря понятиям, описанным выше, можно дать определение вершинной связности графа. *Вершинная связность графа* $\kappa(G)$ – это наименьшее количество вершин графа G, где при удалении данного числа вершин, граф G станет несвязным или тривиальным. Тогда граф $G(V,E)$ по определению можно

назвать *вершинно k-связным графом*, если $\kappa(G) \geq k$, при этом $|V| > k$. Следовательно, при удалении менее, чем k вершин, связность сохраняется.

Свойство k-связности является обобщением свойства связности, при $k = 1$ свойство k-связности эквивалентно обычной связности. Из определения вершинно k-связного графа следует, что $d(v) \geq k$, для любой вершины v данного графа. Далее вершинно k-связный граф будет именоваться, как k-связный граф.

Ребро $e \in E(G)$ называется *существенным*, если при его удалении связность графа уменьшается. Граф называется *минимальным*, если все ребра в нём существенны. Легко видеть, что при удалении любого ребра в минимальном k-связном графе, связность становится равной $k - 1$. Примером минимального 2-связного графа является простой цикл.

Ребро $e \in E(G)$ называется *нестягиваемым*, если при его стягивании k-связный граф не сохраняет k-связность. Тогда k-связный граф называется *минимальным по стягиванию*, если все его ребра являются нестягиваемыми.

1.2. История изучения минимальных и минимальных по стягиванию k-связных графов

В XX веке началось изучение k-связных графов. Наибольший вклад оказали такие ученые, как Менгер, Форд, Фалкерсон, Татт, Уитни, Мадер, Дирак и другие. Сначала были рассмотрены свойства для малых значений k: *1*-связный граф является тривиальным случаем k-связного графа. Достаточно подробно были описаны *2*-связные графы. Д.В.Карпов в своих трудах рассматривал свойства и особенности как *2*-связных графов, так и k-связных графов, критических и многих других.

Было введено понятие «разделяющее множество» – это такое множество вершин $T \subset V(G)$, что при их удалении граф $G - T$ становится несвязным, то есть не все вершины $V \backslash T$ лежат в одной компоненте связности графа $G - T$. *Компонента связности* – это максимальный по включению подграф, являющийся связным. *k-разделяющим множеством* называется такое разделяющее множество, которое содержит ровно k вершин. Поэтому если ребро

(a, b) является нестягиваемым в неполном k-связном графе, то существует k-разделяющее множество, содержащее вершины a, b.

По теореме Менгера для вершинной k-связности: наименьшее число вершин, разделяющих две несмежные вершины a и b, равно наибольшему числу непересекающихся простых $(a - b)$ путей, не имеющих общих вершин.[4] Следовательно, между вершинами a и b существует k вершинно-непересекающихся путей тогда и только тогда, когда после удаления $(k - 1)$ вершины существует путь из a в b.

Другими словами, если связность G равна k, то для каждой пары вершин a и b между ними существует k вершинно-непересекающихся путей.[5] И наоборот, если для каждых двух вершин a и b существует k вершинно-непересекающихся путей, то связность G не меньше k.

Соответственно, можно дать еще одно определение k-связному графу, не используя разделяющее множество: k-связный граф – это граф, где любая пара его вершин соединена по крайне мере k вершинно-непересекающимися путями.

Углубляясь в теорию графов, необходимо установить понятие «разреза». *Разрезом* называется множество, состоящее из вершин и ребер, в котором есть хотя бы одно ребро, и такое, что при удалении этого множества теряется связность. Тогда разрез в k-связном графе – k-разделяющее множество, состоящее из вершин и хотя бы одного ребра. С другой стороны, в теории транспортных сетей *разрезом* называют разбиение множества всех вершин графа $V(\mathrm{G})$ на два подмножества S и T, где $s \in S$, $t \in T$ и $S \cap T = \emptyset$. *Величиной разреза* называется сумма пропускных способностей ребер, которые соединяют вершины из разных разделяющих множеств – $\sum_{s \in S} \sum_{t \in T} c(s, t)$. *Минимальный разрез* – разрез с минимальной пропускной способностью.

Если k-связный граф G минимален, то для каждого ребра e существует разрез, содержащий e и $k - 1$ вершину.

В 1972 году Мадер [6] доказал, что если G – минимальный k-связный граф, то $G - V_k(G)$ является лесом, то есть G_{k+1} является лесом, где G_{k+1} – подграф

графа G на множестве вершин, имеющих степень больше k, а $V_k(G)$ – множество вершин, имеющих степень k. Вследствие этого в 1979 году Мадер [7] вывел нижнюю оценку $v_k(G) \geq \frac{(k-1)\,v(G)+2k}{2k-1}$ для минимального k-связного графа G, где $v_k(G)$ – это количество вершин со степенью k. Данная нижняя оценка является точной, она была выведена из следствий и лемм Мадера. Также результатом исследований Мадера является то, что каждый минимальный k-связный граф имеет не менее $k + 1$ вершины степени k.

Далее общих результатов в этом направлении не было, улучшения были получены только для определенных значений k. У. Татт полностью описал структуру минимального и минимального по стягиванию 3-связного графа. В дополнение одним из важных результатов У. Татта [8] является то, что из любого 3-связного графа с помощью стягивания или удаления ребер можно получить колесо. Пример колеса представлен на рис. 1.1.

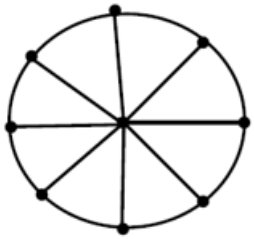

Рис. 1.1. Пример колеса, полученного из 3-связного графа.

После Р. Халин [9] доказал, что в минимальном k-связном графе существуют вершины со степенью k. Существуют экстремальные минимальные k-связные графы – это графы, для которых для любого $k \geq 2$ нижняя оценка Мадера превращается в равенство. С этого момента начались исследования доли вершин степени k в минимальных и минимальных по стягиванию k-связных графах.

Случай с 4-связными графами был рассмотрен М.Фонте [10] и Н.Мартиновым [11]. Было показано, что из минимальности по стягиванию следует минимальность и граф является регулярным, то есть все его вершины

имеют степень 4 и $c_4 = 1$. Здесь c_k – минимальная доля вершин степени k среди вершин минимального и минимального по стягиванию k-связного графа.

Для $k \geq 5$ существуют примеры с одним из свойств, либо минимальность, либо минимальность по стягиванию, поэтому совместное рассмотрение этих свойств вызвало особый интерес. Из утверждения, доказанного К. Андо, К. Каварабаяши [12], что в минимальном по стягиванию 5-связном графе любая вершина будет смежна хотя бы с двумя вершинами степени 5, следует нижняя оценка количества вершин степени 5, равная $\frac{2}{5}$.

В 2010 году К. Андо и Ю. Ченгфу [13] доказали новую нижнюю оценку. Используя свойство Ж. Су [14]: каждая вершина минимального и минимального по стягиванию 5-связного графа имеет двух соседей, имеющих степень 5. И свойство К. Андо и Т. Иваси [15], что если в минимальном и минимальном по стягиванию 5-связном графе существует вершина, степень которой не равна 5, но имеющая две соседние смежные вершины, степени которых равны 5, то существует третья соседняя вершина, имеющая степень 5. Отсюда было получено, что доля вершин степени 5 составляет по крайней мере $\frac{1}{2}$ количества всех вершин графа.

С. А. Образцова и В. А. Пастор [21] вывели результаты для минимальных и минимальных по стягиванию k-связных графов: верхнюю оценку количества вершин степени 5: $c_5 < \frac{17}{22}$, и нижнюю оценку – $\frac{4}{7}$. На сегодняшний момент лучшей нижней оценкой считается $\frac{2}{3}$.

Лучшей оценкой доли вершин степени 6 в минимальном по стягиванию 6-связном графе является $|V_6| \geq \frac{1}{7}|G|, V_6$ – это вершины со степенью равной 6. Для минимального по стягиванию 7-связного графа Ж. Су, Кс. Юан, М. Ли [16] доказали, что нижняя оценка равна $|V_7| \geq \frac{1}{22}|G|$. На данный момент еще не доказано существование хотя бы одной вершины степени 8 в минимальном и минимальном по стягиванию 8-связном графе. При $6 \leq k \leq 10$ в минимальных и минимальных по стягиванию k-связных графов С. А. Образцова и В. А. Пастор

доказали строгую нижнюю оценку равную $v_k(G) > \frac{1}{2}v(G)$. Как оказалось, чем больше k, тем труднее находить долю вершин степени k.

Сложность в изучении k-связных графов при $k \geq 2$ состоит в том, что разделяющие множества, состоящие из $k \geq 2$ вершин, могут быть зависимыми. В данном случае разделяющие множества S, T называются независимыми, если S не разделяет T и T не разделяет S. До сих пор неизвестны все возможные k-связные, минимальные и минимальные по стягиванию k-связные графы при $k \geq 5$ и большом числе узлов.

На сегодняшний день существует множество алгоритмов, описывающих определенное построение k-связных графов. Однако в данной работе будет рассматриваться не построение таких графов, а их поиск внутри заданного или полного графа.

ГЛАВА 2. ОБЗОР МЕТОДОВ

С развитием интереса к k-связным графам, так же вырос «спрос» на алгоритмы, связанные с ними. С теоретической точки зрения были выведены полезные результаты (теоремы, утверждения, следствия) при изучении k-связных графов, поэтому исследователи стали рассматривать свойства данных графов с другой, алгоритмической, программной стороны. Начали разрабатываться алгоритмы построения k-связного графа, потому что до этого построить такой граф, где $k > 4$, было возможно только теоретически. Данные алгоритмы улучшались с течением времени: уменьшалась алгоритмическая сложность и затраты на выполнение алгоритма, использовались различные свойства графов. Также требовались и другие алгоритмы. В частности, алгоритмы проверки, с помощью которых можно понять, является ли граф k-связным.

2.1. Алгоритм Эвена

Вспомним, что k-связный граф – это граф, который при удалении менее k вершин остается связным. Для определения является ли граф k-связным, можно осуществить проверку, что $\kappa(G) \geq k$. Тогда нужно иметь алгоритм нахождения вершинной связности $\kappa(G)$.

В 1975 году С. Эвен опубликовал статью [24], в которой представленный алгоритм предназначался для проверки k-связности, графа на n вершинах. Сначала в статье обсуждался случай неориентированных графов, затем было доказано, что этот вариант работает и для ориентированных, но здесь будет описан только нужный случай – неориентированный.

Множеством, разделяющим вершины a и b, называется множество вершин S, если $\{a, b\} \subseteq V - S$ и каждый путь между вершинами a и b проходит хотя бы через одну вершину из множества S.

Через $N^G(a, b)$ обозначается связность между вершинами a и b в графе G. Она определяется следующим образом:

— если $(a, b) \in E$, то равна $|V| - 1$;

— иначе принимает значение мощности наименьшего множества, разделяющего вершины a и b.

Благодаря теореме Менгера и связи между связностью и сетевым потоком, можно утверждать, что алгоритм Форда-Фалкерсона может использоваться для определения связности графа. Теорема о максимальном потоке и минимальном разрезе обуславливает, что максимальное количество вершинно-непересекающихся путей, соединяющих вершины a и b, равняется минимальной мощности множества, разделяющего вершины a и b, если между ними отсутствует ребро; иначе количество путей на единицу больше наименьшего числа вершин множества, отделяющего a от b после того, как ребро между ними было удалено.

Таким образом, для каждой пары вершин нужно найти максимальное количество вершинно-непересекающихся путей, тогда минимальное значение для всех пар и будет связностью: $\kappa(G) = \min_{a,b \in V} N^G(a,b)$.

Для каждой пары вершин строится потоковая сеть с $2|V|$ вершинами и $2|E| + |V|$ ребрами. Пропускная способность у всех ребер равна единице. Каждый путь между двумя вершинами ищется за O($|E|$) шагов. Поскольку связность может достигать $|V| - 1$, то вся процедура нахождения максимального числа вершинно-непересекающихся путей, соединяющих одну пару вершин, обходится O($|V||E|$) или O($|V|^3$), так как O($|E|$) $\leq$ O($|V|^2$). Таким образом, повторение данного действия для всех пар будет стоить O($|V|^5$).

Это занимает достаточно ресурсов, поэтому С. Эвен предложил усовершенствованный алгоритм. Данный алгоритм основан на алгоритме Клейтмана, который занимает O($k^2|V|^3$) шагов. С. Эвен предложил алгоритм, который теперь требует O($k|V|^3 + k^3|V|^2$) шагов. В статье [11] было утверждение, что поиск в глубину, который использовался в алгоритмах проверки для маленьких $k = 1, 2, 3$ не актуален для более больших значений k, поэтому здесь не используется.

Пусть $V = \{v_1, v_2, \dots, v_n\}$, $L_j = \{v_1, v_2, \dots, v_{j-1}\}$, $\widetilde{G_j}$ состоит из графа G, но с добавляем новой вершины s, которая соединяется со всеми вершинами из L_j, $\widetilde{G_j}$ называется вспомогательным (расширенным) графом.

Лемма 1. Если в G каждая вершина v_i $(1 \leq i < j)$ может быть соединена с u $(u \in V, u \notin L_j)$ через k вершинно-непересекающиеся пути, то в $\widetilde{G_j}$ существует k вершинно-непересекающихся путей между s и u. [24]

Пусть вершина v_j – вершина с наименьшим номером графа G, что для некоторой вершины v_i, где $i < j$, не существует k вершинно-непересекающихся путей, соединяющих v_i и v_j в G.

Лемма 2. Не существует k вершинно-непересекающихся путей, соединяющих s и v_j в $\widetilde{G_j}$. [24]

Алгоритм Эвена:

— Проверяем, что $N^G(v_i, v_j) \geq$ k, $1 \leq i \leq j \leq k$, если нет, то выход: $\kappa(G) < k$;

— проверяем $N^{\widetilde{G_j}}(\mathrm{s}, v_j) \geq$ k, при $k+1 \leq j \leq |V|$, если нет, то выход: $\kappa(G) < k$;

— выход: $\kappa(G) \geq k$.

Данный алгоритм применим, поскольку если $\kappa(G)$ как минимум равняется k, то по лемме 1 на шаге 2 не будет обнаружено ошибок, и алгоритм остановится с верным ответом. Если $\kappa(G) < k$, то по лемме 2 произойдет сбой, и алгоритм снова остановится с правильным ответом.

Вычислительная сложность первого пункта алгоритма составляет не более O($k^3|E|$) шагов, а второго пункта – не более O(k$|V||E|$) шагов. Таким образом, для всего алгоритма требуется O($k^3|E| + k|V||E|$) шагов. Поскольку O($|E|$) $\leq |V|^2$, то количество шагов в алгоритме Эвена ограничено O($k^3|V|^2 + k|V|^3$). Данный алгоритм является усовершенствованной версией алгоритма Клейтмана. Экономия в данном алгоритме достигается добавлением новой вершины s и

проверки ее связности с вершиной j в $\widetilde{G_j}$, вместо повторяющейся проверки k различных вершин.

2.2. Алгоритм Форда-Фалкерсона

До сих пор не было сказано, как проверить, что $N^G(a,b) \geq k$. Данное условие, что $N^G(a,b) \geq k$ выполняется тогда и только тогда, когда значение максимального потока большое или равно k. А значение максимального потока можно найти с помощью алгоритма Форда-Фалкерсона или алгоритма Диница, например.

Для дальнейшего разбора этих алгоритмов нужно ввести новые определения.

Поток – это функция $f: V^2 \longrightarrow \mathbb{R}$ со свойствами для любых вершин u и v:

— $f(u,v) \leq c(u,v)$;

— $f(u,v) = -f(v,u)$;

— $\sum_{w \in V} f(u,w) = 0$, для всех $u \in V$, кроме s и t.

где $c(u,v) \geq 0$ – это пропускная способность, а s и t – это исток и сток, соответственно.

Тогда *величиной потока* называют $|f| = \sum_{v \in V} f(s,v) = \sum_{w \in V} f(w,t)$. *Остаточная пропускная способность* $c_f(u,v) = c(u,v) - f(u,v)$. А *остаточная сеть* – граф $G_f = (V, E_f)$, при этом E_f – множество ребер, где $c_f(u,v) > 0$. И *дополняющий путь* – это путь $(u_1, u_2, \ldots, u_k)$ в остаточной сети, где $u_1 = s$, $u_k = t$, $c_f(u_i, u_{i+1}) > 0$. Доказано, что поток максимален тогда и только тогда, когда не существует дополняющего пути в остаточной сети. [25]

Алгоритм Форда-Фалкерсона:

— обнуляем все потоки. Остаточная сеть совпадает с изначальной;

— в остаточной сети находим любой путь из истока в сток. Если такого пути нет, то останавливаемся;

— пускаем поток через этот путь:

— на этом пути в остаточной сети ищем ребро с минимальной пропускной способностью c_{min};

— для каждого ребра на этом пути увеличиваем поток на c_{min}, противоположному ребру уменьшаем на c_{min};

— вычисляем для ребер на этом пути новую пропускную способность. Если ненулевая, то оставляем ребро в остаточной сети; если нет, то удаляем.

В алгоритме нет конкретного пути, который находится в шаге 2. Из-за этого алгоритм будет сходится только при целых значениях пропускной способности, при этом для больших значения может работать достаточно долго. Можно искать кратчайший путь алгоритмом Диница.

2.3. Алгоритм Диница

В алгоритме Диница происходит поиск максимального потока.

В 1975 году С. Эвен и Э. Тартьян в своей статье [26] представили алгоритм, аналогичный алгоритму Диница, за $O(|V|^2|E|)$, эта версия была независимо открыта С. Эвеном и Дж. Хопкрофтом.

Алгоритм выполняется поэтапно (по фазам), число фаз не более $|V| - 1$. На каждой фазе величина максимального потока увеличивается. Доказательство максимальности такое же, как у Форда-Фалкерсона. При этом авторы утверждают, что ребро можно использовать в обоих направлениях. Следовательно, если $f(u, v) \leq c(u, v)$, то движение в прямом направлении; иначе, если $f(u, v) > 0$, то движение в обратном направлении.

Как известно С. Эвен и Э. Тартьян дополнили алгоритм Диница, чтобы было возможно найти значение связности между вершинами a и b – $N^G(a, b)$. Изначальный алгоритм Диница представлял решение классической задачи о нахождении максимального потока, где используется только пропускная способность ребер. Однако в данном случае важным фактором является ограничение пропускной способности вершин (вес вершины), так как в алгоритме Эвена производится проверка на вершинную, а не реберную k-связность. Благодаря следующему преобразованию – «расширению» графа – можно свести решение второй задачи к решению первой и применить алгоритм Диница.

Перед началом алгоритма нужно преобразовать неориентированный граф, чтобы все ребра стали ориентированными в оба направления. Затем граф «расширяется»: все вершины, кроме s и t разделяются на 2. В первую вершину направлены входящие ребра, из второй вершины направлены выходящие ребра и эти две вершины соединяются один ребром (от первой ко второй) с весом вершины $c(e) = c(v)$. На рис. 2.1 приведен пример разделения вершин на 2.

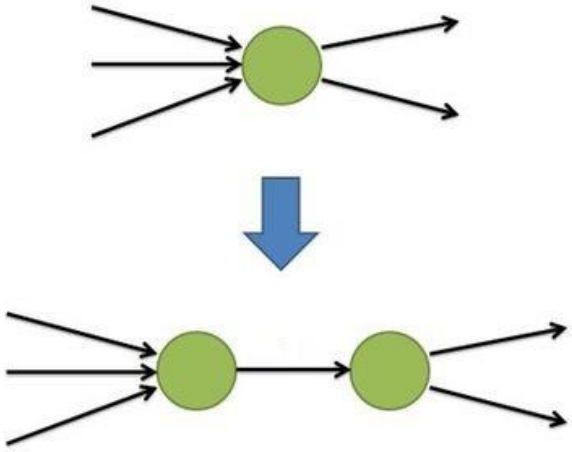

Рис. 2.1. Пример разделения вершины.

В начале алгоритма происходит построение вспомогательного графа, поскольку в первый раз остаточная сеть G_f равна исходному графу. Вспомогательный граф G_L – граф G_f, в котором остались ребра (u, v), где $d[v] = d[u] + 1$, $d[v]$ – «уровень» вершины v, то есть количество ребер в кратчайшем пути между вершинами s и v. Построение вспомогательного графа осуществляется с помощью поиска в ширину. Изначально идет присваивание истоку – вершине s – «уровень» 0, то есть $d[s] = 0$. Затем все вершины v достижимые из s через единственное ребро имеют «уровень» 1, то есть $d[v] = 1$. Далее вершины v обозначаются за u, как пройденные вершины, и ищутся следующие вершины v, достижимые из вершин u, за одно ребро, то есть должно быть $d[v] = d[u] + 1$. Если переход случился в вершину, уже имеющую «уровень», то есть либо назад, либо на этом же «уровне» $d[v] \leq d[u]$, то данный путь не учитывается, поскольку идет поиск кратчайшего пути из s в t. Данная

процедура построения вспомогательного графа происходит за O($|E|$). Все пути от s до t имеют, соответственно, длину $d[t]$. На рис. 2.2 приведен пример вспомогательной сети. Пунктирные ребра не входят в данный граф.

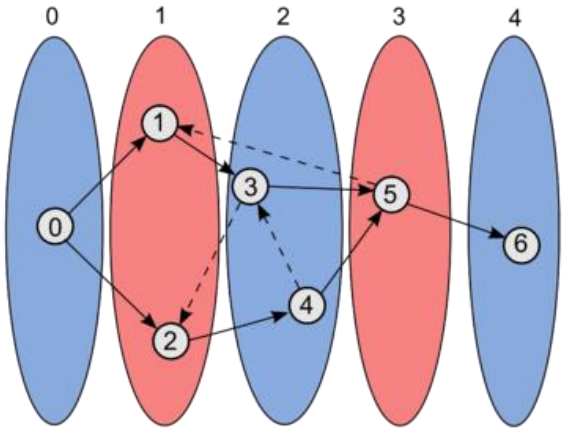

Рис. 2.2. Пример вспомогательной сети.

Далее пути, найденные во вспомогательном графе, используются для нахождения дополняющих путей с помощью поиска в глубину.

Рассматриваются пути из s в t, где для любого ребра (u, v): $d[v] = d[u] + 1$. По такому пути пропускается максимальный поток. В нашем случае пропускные способности всех ребер равны 1. Затем в пройденном направлении вычитается величина потока, а в обратном прибавляется. Если из вершины v нет никакого дальнейшего пути в $d[v] + 1$, то ребро, по которому пришли в вершину v, удаляется из пути и вспомогательного графа. А дальше происходит возвращение в вершину, предшествующую v, и продолжается поиск оттуда.

Поиск одного успешного пути, не учитывая возвращения, занимает не более O($d[t]$). Таким образом, общее время работы одной фазы ограничено O($|V||E|$). А общее количество фаз не превышает $|V| - 1$, тогда весь алгоритм занимает O($|V|^2|E|$).

Таким образом, алгоритм Диница можно разделить на следующие этапы:

— обнуляем поток $f(e) = 0,\ e \in E$;

— строим вспомогательную сеть G_L из остаточной сети G_f. Перед построением вспомогательной сети производится построение остаточного графа G_f. Изначально G_f является исходным графом. Если $d[t] = \infty$, то останавливаемся;

— находим блокирующий поток f' во вспомогательном графе G_L. Блокирующий поток f' – такой поток, что любой $s - t$ путь содержит насыщенное этим потоком ребро, то есть хотя бы у одного ребра $c(e) = 0$. Блокирующий поток можно найти двумя способами:

— искать $s - t$ пути по одному, пока такой путь есть;

— аналогично предыдущему пункту, но при поиске удалять ребра, через которые невозможно дойти до стока;

— дополняем наш поток f блокирующим потоком f'.

Таким образом, вычислительная сложность рассмотренных выше алгоритмов равна:

— Форда-Фалкерсона: O($k^2 + |V|$)

— Диница: O($|V|^2|E|$), O($|E|^{\frac{2}{3}}$) для единичных сетей. Единичная сеть – это сеть, где пропускная способность любого ребра равна единице. O($|V|\sqrt{|E|}$) – если в единичной сети у каждой вершины, кроме истока и стока, есть либо ровно одно входящее ребро, либо ровно одно исходящее;

— Эвена: O($k^3|E| + k|V||E|$).

2.4. Улучшенный алгоритм Галила

В 1980 Галил опубликовал алгоритм [27] нахождения вершинной связности графа со сложностью O($min(\kappa, |V|^{\frac{2}{3}})|V||E|$). Также в 1987 году Г. Милинд и С. Милинд в своей статье [28] описали реализацию самого быстрого и известного, на тот момент времени, алгоритма нахождения вершинной связности графов. За основу в описанном алгоритме берется алгоритм Галила, поскольку он являлся более эффективным, и в своей статье Г. Милинд и С. Милинд

улучшают результат данного алгоритма не во временных затратах, а в затратах памяти O($|V||E|$).

Когда вершинная связность $\kappa(G)$ мала, то существуют линейные алгоритмы, относительно числа ребер, для детерминированной связности: $\kappa(G) = 2, 3, 4$. Для проверки k-связности, где $\kappa(G) \geq k$, имеются алгоритмы Эвена, Эвена-Тартьяна, Галила с алгоритмической сложностью O($|V||E|$). В данной статье рассматривается случай, когда $\kappa(G)$ велико. Известные на тот момент детерминированные алгоритмы используют метод поиска максимального потока в единичных сетях.

В алгоритме Галила вместо проверки, что $N^G(v_i, v_j) \geq k$, можно выполнить другие действия, которые приведут к такому же результату и не изменят временную сложность алгоритма.

Алгоритм:

— инициализация: $k = 1, MIN = |V| - 1$;

— найдем $N^G(v_i, v_k)$, $1 \leq i \leq k-1$;

— обновим MIN: $MIN = min(\min\limits_{1 \leq i \leq k-1} N^G(v_i, v_k),\ MIN)$;

— если $MIN < k$, то Выход: $\kappa(G) = MIN$;

— проверяем $N^{\widetilde{G_j}}(s, v_j) \geq k$, $k+1 \leq j \leq |V|$, если нет, то выход: $\kappa(G) = k - 1$;

— $k++$, повторить с шага 2.

Используя алгоритм Диница, можно сократить вычислительную сложность шагов 2 и 5.

Вычислительная сложность:

— шаг 2: O($k|E||V|^{\frac{1}{2}}$);

— шаг 5: O($|V||E|$);

— весь алгоритм: O($\kappa(G)^2|E||V|^{\frac{1}{2}} + \kappa(G)|V||E|$) = O($\kappa(G)|E||V|^{\frac{1}{2}}(\kappa(G) + |V|^{\frac{1}{2}})$) = O(max($\kappa(G)$, $|V|^{\frac{1}{2}}$)$\kappa(G)|E||V|^{\frac{1}{2}}$).

2.5. Используемые алгоритмы

Из всех рассмотренных в предыдущих параграфов алгоритмов, связанных с проверкой k-связности графа, для программной реализации был выбран алгоритм Эвена и алгоритм Диница из статей С. Эвена и Э. Тартьяна [11, 12]. Усовершенствованный алгоритм Галила [14] не подходит под данную проверку, так как он предназначен для поиска связности графа в общем. А в алгоритме Эвена определяется более простая задача: верно ли, что $\kappa(G) \geq k$ для известных G и k.

Для проверки, является ли k-связный граф минимальным, было решено воспользоваться свойствами, что граф G является минимальным k-связным графом тогда и только тогда, когда любое ребро входит в разрез. Как известно, при удалении ребра e и $k - 1$ вершин образуется вершинно-реберный разрез. Тогда, при удалении ребра и вершин разреза, вершины, инцидентные ему, будут лежать в разных компонентах связности. Таким образом, будет достаточно проверить наличие k-связности между этими вершинами. Следовательно, не нужно проверять весь граф на k-связность, а достаточно применить алгоритм Диница к двум вершинам, инцидентных удаляемому ребру. Если максимальный поток между этими вершинами будет больше либо равен k, то такой граф не будет являться минимальным.

Проверка на минимальность по стягиванию k-связных графов будет осуществляться следующим образом:

— сначала происходит удаление двух вершин, которые являются концами стягиваемого ребра, и инцидентных им ребер;

— затем получившийся новый граф поступает на проверку $(k-1)$-связности;

— если получившийся граф является $(k-1)$-связным, то исходный граф не является минимальным по стягиванию, и следует остановка проверки. Иначе, проверка продолжается.

Таким образом, весь алгоритм поиска минимальных и минимальных по стягиванию k-связных графов внутри поступающего на вход или заданного полного графа выглядит следующим образом:

— задается граф с n вершинами, также задается k;

— осуществляется перебор графов с их проверкой на k-связность: проверяется граф, удаляется ребро, очередная проверка, удаление ребра и так далее, пока степень вершин $>\ k$. Затем идет возвращение на одно ребро назад и проверяется «другой путь» удаления ребер;

— найденные k-связные графы проверяются на минимальность. Поочередно удаляются ребра, и запускается алгоритма Диница. Если найден поток, который $\geq\ k$, то данный граф точно не является минимальным, значит можно перейти к проверке следующего графа;

— найденные минимальные k-связные графы проверяются на минимальность по стягиванию. Поочередно происходит стягивание ребер, если получившийся граф будет являться $(k-1)$-связным, то исходный проверяемый граф не является минимальным по стягиваю, значит можно перейти к проверке следующих графов.

ГЛАВА 3. ПРОГРАМНАЯ РЕАЛИЗАЦИЯ

В ходе работы был программно реализован алгоритм перебора графов и проверки графов на свойства:

— k-связности;

— минимальности;

— минимальности по стягиванию.

3.1. Инструментальные средства

В процессе реализации были созданы следующие классы:

- $class\ Matrix$ содержит матрицу смежности графа и операции связанные с ней;
- $class\ Graph$ содержит всю информацию о графе и алгоритмы, связанные с ним.

Связь данных классов можно представить в виде диаграммы классов. Их построение делается с помощью унифицированного языка моделирования (UML). На UML диаграмме класс обозначается прямоугольником, разделенным на три части:

- имя класса;
- объекты (поля) класса;
- методы (функции) класса.

Для объектов и методов существуют спецификаторы доступа (видимости). В данной программе используются $public$ (публичный) и $private$ (приватный). Соответствующий спецификатор имеет перед полем или функцией знак «+» или знак «–».

Диаграмма классов представлена на рис. 3.1. На ней изображены логически важные функции.

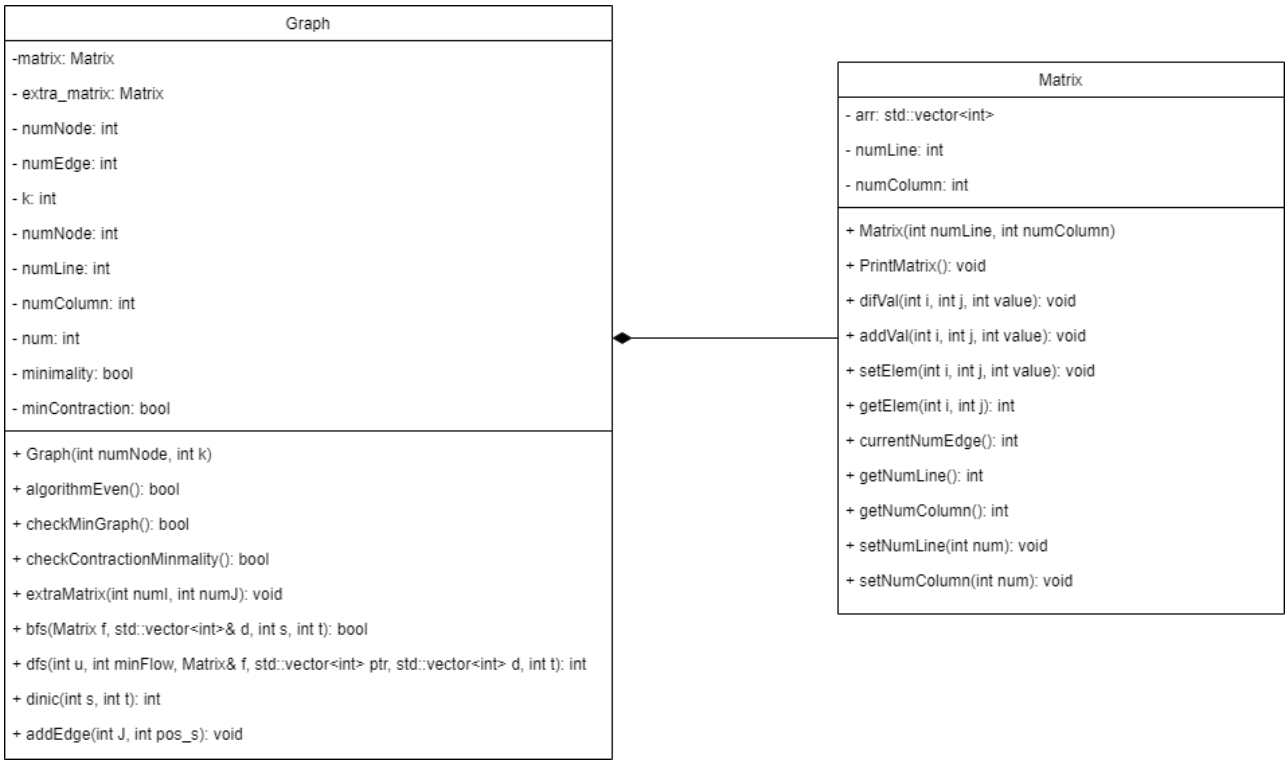

Рис. 3.1. Диаграмма классов программы.

3.2. Работа основных методов

Для реализации данной задачи необходимо рассмотреть работу основополагающих методов. В функции $algorithmEven$ происходит реализация алгоритма Эвена, который рассматривался в предыдущей главе.

Блок-схема данного метода приведена на рис. 3.2.

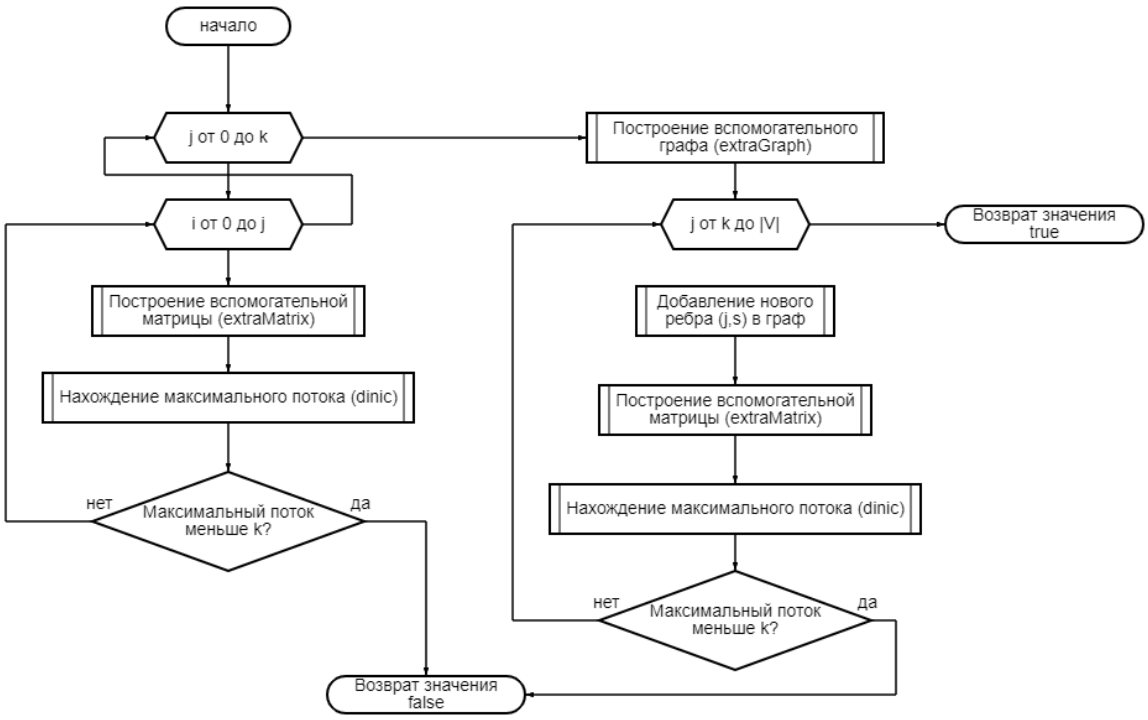

Рис. 3.2. Блок схема метода algorithmEven.

В приведенной блок-схеме на рис. 3.2 построением вспомогательной матрицы ($extraMatrix$) является операция «расширения» графа. Данная операция нужна для того, чтобы свести задачу о пропускной способности вершин к задаче о пропускной способности ребер в алгоритме Эвена. Данное введение позволяет использовать классический алгоритм Диница для заданной задачи.

А в методе построения вспомогательного графа ($extraGraph$) производится построение вспомогательного графа $\widetilde{G_J}$.

Функция поиска величины максимального потока ($dinic$) осуществляется на основе алгоритма Диница. Рассмотрим блок-схему функции $dinic$,

реализующую алгоритм Диница, на рис. 3.3. Он используется в алгоритме Эвена и так же был описан в предыдущей главе.

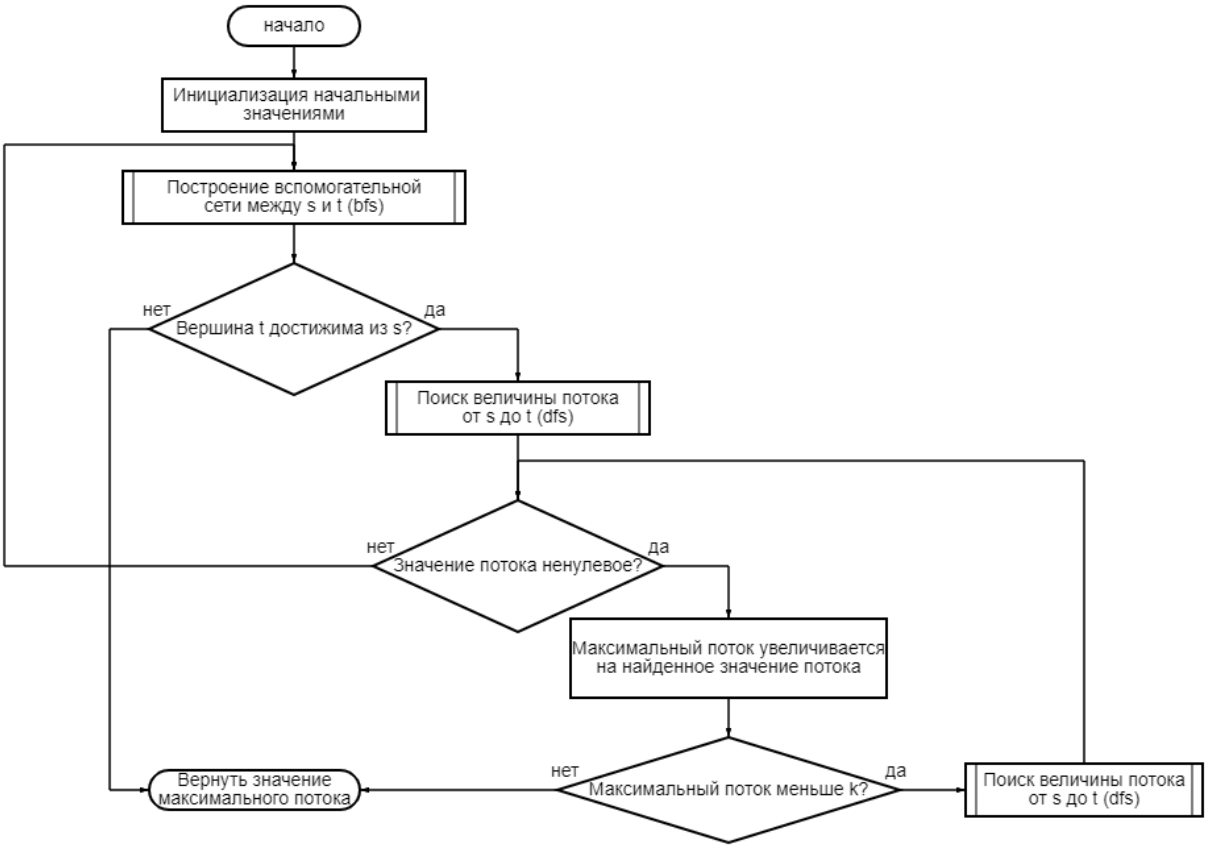

Рис. 3.3. Блок схема метода dinic.

Рассматриваемый алгоритм реализует построение вспомогательной сети и поиск величин потока от истока s до стока t. Результатом работы данного метода является получение величины максимального потока между s и t.

Поиск в ширину (bfs) осуществляет построение вспомогательной сети, то есть вычисление «уровней» вершин $d[v]$, где $v \in V(G)$. Начинается обход с истока – s, он помещается в вектор рассматриваемых вершин Q. Затем, пока вектор Q не пуст, производится вычисление «уровней» вершин. В качестве вершины u выбирается первая вершина, лежащая в векторе Q. Затем, проходя по всем остальным вершинам $v \in V(G)$, производится проверка:

— если вершина v еще не имеет «уровня» и у ребра, соединяющего u и v, осталась пропускная способность, то данная вершина добавляется в вектор Q и $d[v] = d[u] + 1$;

— иначе осуществляется переход к рассмотрению следующей вершины.

Поиск в глубину (dfs) величины потока осуществляется следующим образом:

— проход осуществляется по вершинам, у которых $d[v] = d[u] + 1$;

— рекурсивно находится значение потока, который проходит от s до t;

— если значение положительное, то из данного ребра «потоковой» матрицы и матрицы смежности вычитается это значение в прямом направлении, и добавляется в обратном направлении. Затем возвращается значение найденного потока;

— если достигнута вершина t, то есть $u = t$ или поток нулевой, то сразу происходит возврат значения потока.

Можно заметить, что в методе $algorithmEven$ используются еще две значимые функции: построение «расширенного» графа в $extraMatrix$ и вспомогательного графа $\widetilde{G_j}$ в $extraGraph$.

В методе $extraMatrix$ осуществляется добавление новых вершин и «перезапись» значений в новой матрице смежности. Исходная вершина становится «входной», а новая, добавленная вершина – «выходной». Между ними также устанавливается ребро.

В методе $extraGraph$ производится добавление новой вершины s к имеющемуся графу. Далее новая вершина соединяется со всеми узлами из L_j, где $L_j = \{v_1, v_2, \ldots, v_{j-1}\}$. Это означает, что в новой матрице смежности устанавливаются единицы в местах пересечения столбца s и строк $j - 1$, и наоборот строки s и столбцов $j - 1$. Затем на каждом шаге L_j будет расширяться на одну вершину, так как в $algorithmEven$ переменная j меняется в цикле от k до $|V|$. Для вспомогательного графа будет достаточно добавлять данную связь новой вершины v_{j-1} с s.

Для проверки свойства минимальности у k-связного графа была реализована функция $checkMinGraph$. Блок-схема функции приведена на рис. 3.4.

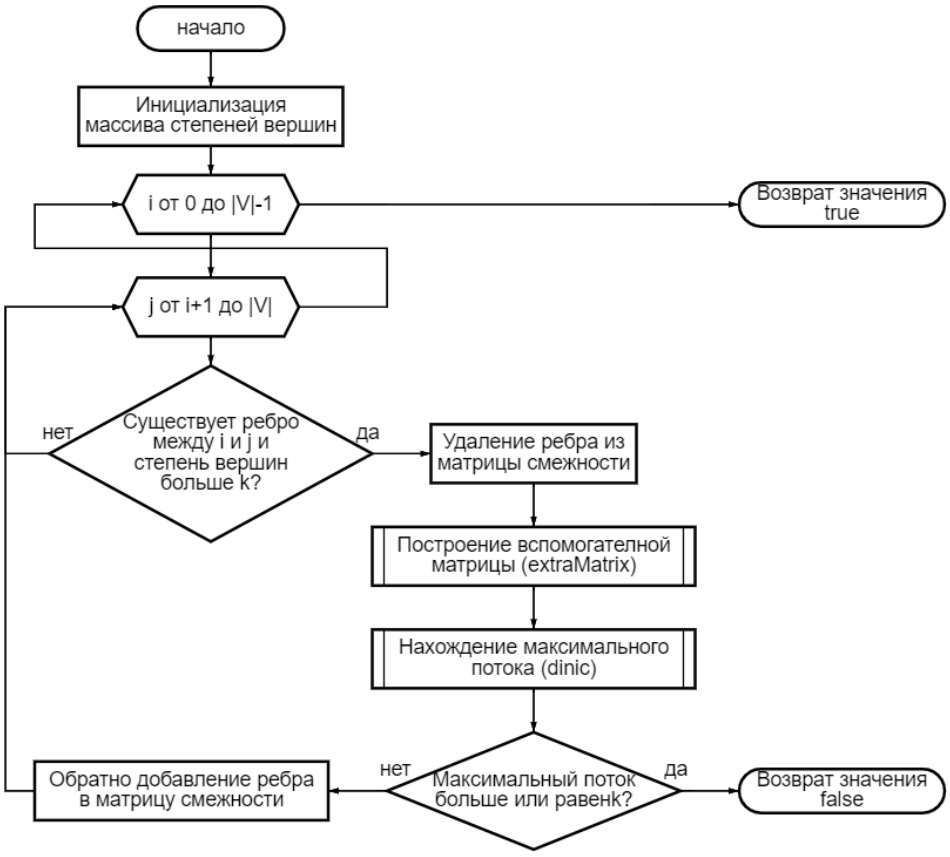

Рис. 3.4. Блок схема метода checkMinGraph.

Из-за того, что все графы являются неориентированными, матрица смежности симметрична относительно главной диагонали. Поэтому, при рассмотрении пары вершин, выбор второй вершины можно осуществлять начиная со следующего номера после номера первой вершины. Данное действие предотвращает рассмотрение одних и тех же случаев дважды. Также цикл по строкам можно осуществлять только до $|V| - 1$, так как последняя вершина уже пройдет все проверки в предыдущих этапах.

Для проверки свойства минимальности по стягиванию у k-связного графа была реализована функция $checkContractionMinimality$.

Блок-схема функции приведена на рис. 3.5.

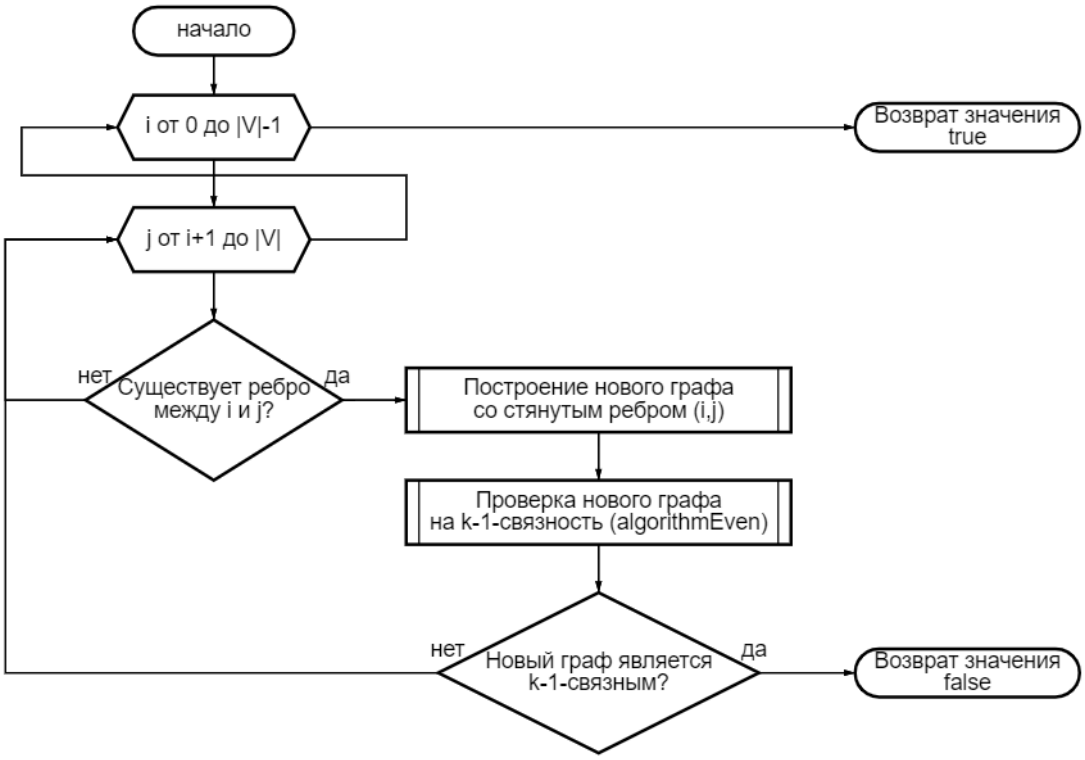

Рис. 3.5. Блок схема метода checkContractionMinimality.

Циклы, аналогичные циклам в методе $checkMinGraph$, присутствуют в данном методе $checkContractionMinimality$ по той же причине.

Благодаря методам, описанным выше, можно проверить является ли граф минимальным и минимальным по стягиванию k-связным. При этом следует заметить, что нахождение таких графов является возможным. Данный поиск реализуется с помощью рекурсивного перебора.

В функции fun осуществляется перебор графов и их проверка на k-связность.

Блок-схема этой функции изображена на рис. 3.6.

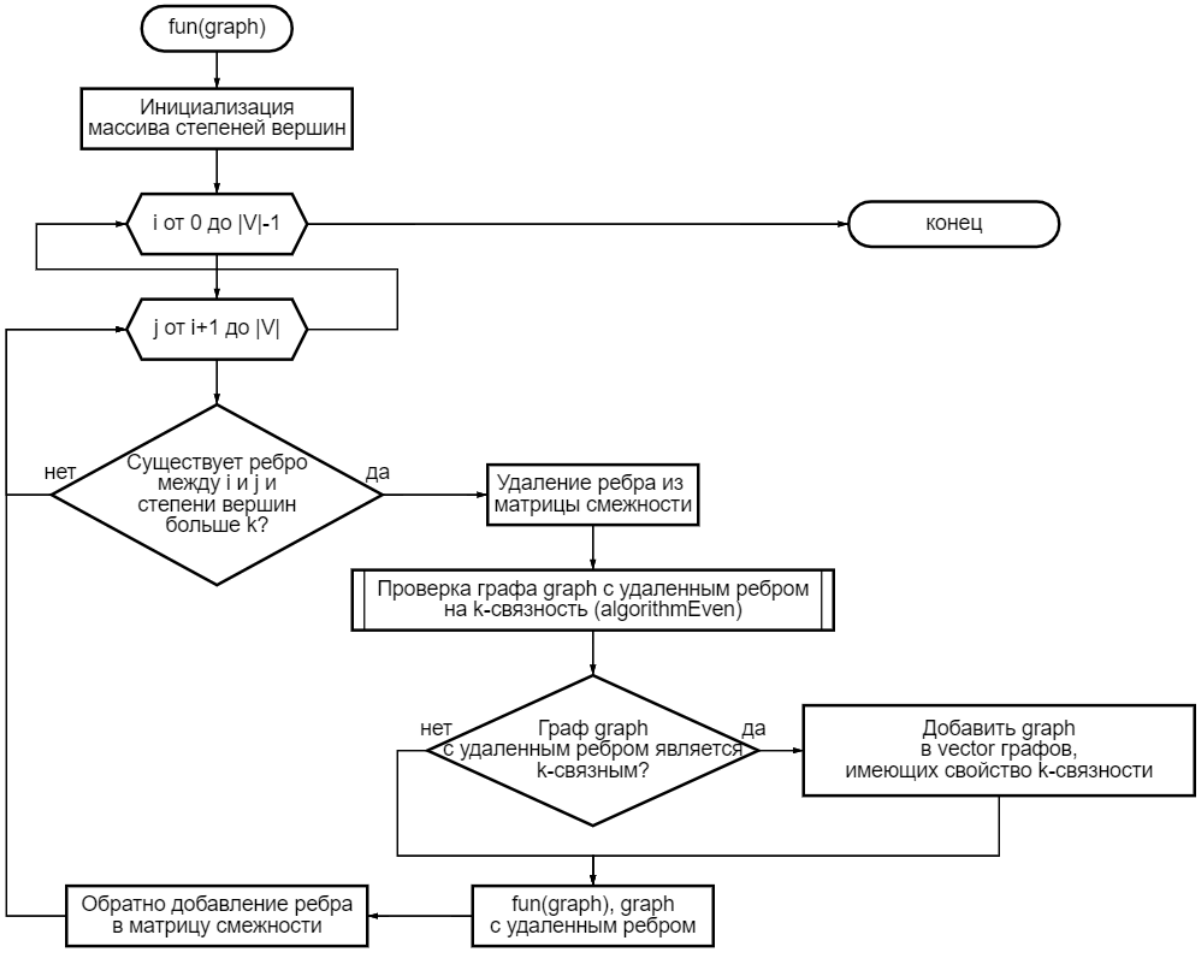

Рис. 3.6. Блок схема метода fun.

Для того, чтобы не создавать новый граф для проверки каждый раз, производится удаление и восстановление ребра в том же графе. Данная процедура производится «перезаписыванием» пропускных способностей ребер в матрице смежности. Таким, образом, осуществляется перебор всевозможных графов, пока существуют ребра и степени вершин больше k. Аналогично предыдущим двум методам в функции fun при выборе ребра вторая вершина начинается со следующего номера после номера первой вершины. И последняя вершина исключается из рассмотрения в ребрах «в роли» первой вершины.

Следует заметить, что для ускорения работы программы рекурсию в функции fun можно раскрыть. Раскрытие рекурсии осуществляется в методе $enumeration\_graph$.

А блок-схема метода $enumeration\_graph$, который осуществляет перебор графов без рекурсии, приведена на рис. 3.7.

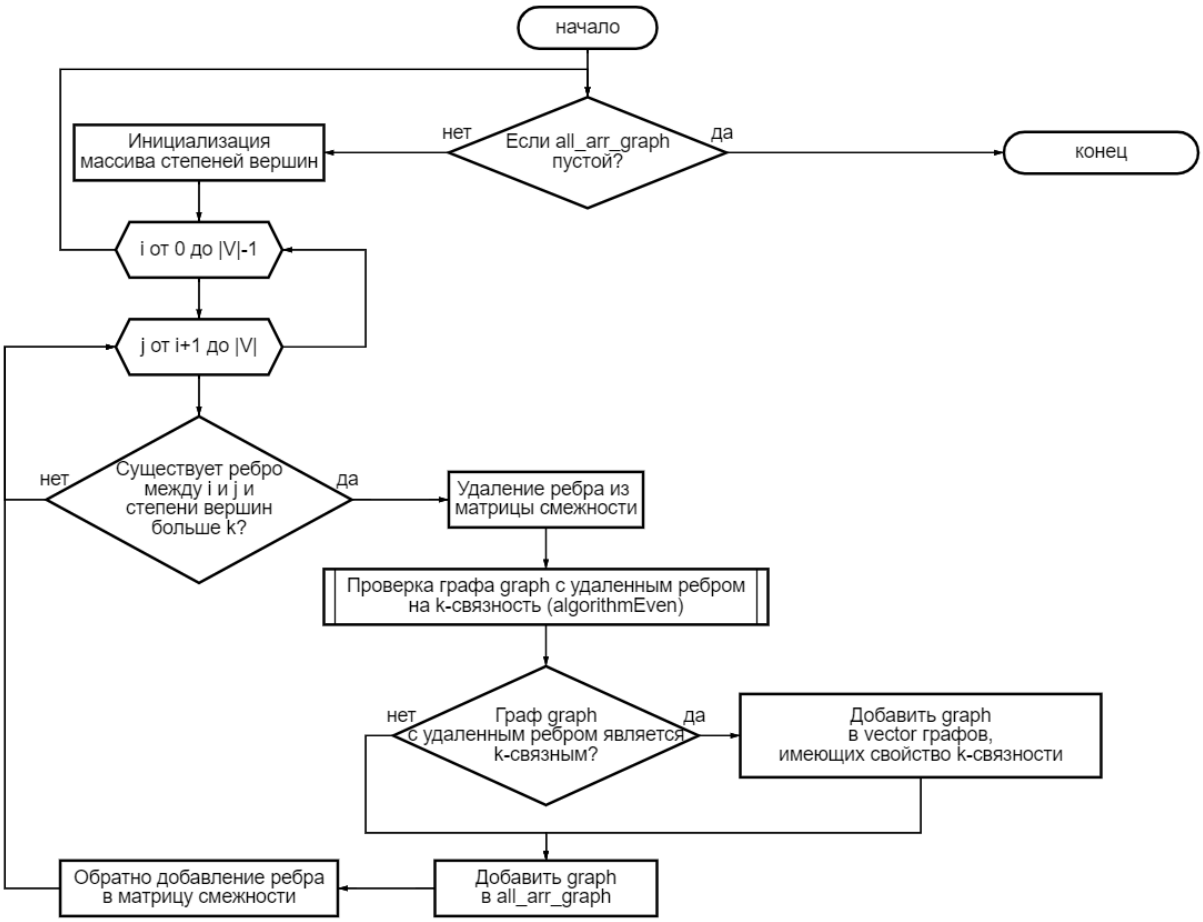

Рис. 3.7. Блок схема метода enumeration_graph.

В рассматриваемой функции переменная all_arr_graph является вектором графов, которые нужно исследовать. Иными словами, туда помещаются графы, у которых перебираются ребра и проверятся наличие k-связности.

Помимо этого, была реализована программа визуализации графов на языке программирования Python с помощью пакета NetworkX. NetworkX – пакет Python для создания, управления и изучения структуры, динамики и функций сложных сетей.

ГЛАВА 4. РЕЗУЛЬТАТЫ И АНАЛИЗ

Одним из важных этапов разработки является получение конечных результатов и их анализ.

Для проведения эксперимента на вход программы поступали полные графы, для которых было замерено время работы каждой проверки на k-связность, минимальность и минимальность по стягиванию.

Сначала была применена оптимизация с построением вспомогательного графа $\widetilde{G_J}$ в методе $extraGraph$. Теперь $\widetilde{G_J}$ строится не каждый раз при новых вершинах, а только в самый первый раз, после чего идет добавление новых ребер с помощью метода $addEdge$.

Следующим этапом является добавление элемента оптимизации:

— в проверке на минимальность в случае, если максимальный поток больше или равен k, то данный граф уже не является минимальным;

— в проверке на минимальность по стягиванию если граф, полученный стягиванием ребра, является $(k-1)$-связным, то изначальный граф точно не является минимальным по стягиванию.

Результаты времени проверок приведены в табл. 4.1.

Таблица 4.1

Время работы проверок для полных графов

N / **time**	**N = 30**	**N = 40**	**N = 50**	**N = 100**
t_k	60мс	137мс	320мс	2756мс
$t_{min}\ (false)$	1мс	1мс	2мс	3мс
$t_{minContr}\ (false)$	12мс	26мс	35мс	199мс
t	73мс	164мс	356мс	≈ 3с

В табл. 4.1 N – количество вершин в графе, t_k – время работы проверки на k-связность, t_{min} – время работы проверки минимальности, $t_{minContr}$ – время

работы проверки минимальности по стягиванию, t – общее время работы всех проверок.

Дополнительно были произведены эксперименты с регулярными графами. Результаты работы с регулярными графами приведены в табл. 4.2.

Таблица 4.2

Время работы проверок для регулярных графов

time / N	t_k	t_{min} $(false)$	$t_{minContr}$ $(false)$	t
N = 20, k = 4	16мс	1мс	11мс	28мс
N = 30, k = 5	40мс	1мс	27мс	68мс
N = 30, k = 5	32мс	1мс	22мс	57мс
N = 40, k = 5	64мс	2мс	72мс	138мс
N = 100, k = 5	1088мс	4мс	683мс	1775мс

На рис. 4.1 представлены исследуемые графы.

Первым изображен граф с 20 вершинами, где соединены каждая четвертая вершина, во втором графе с 30 вершинами, соединены каждая третья вершина. В следующем графе, так же имеющем 30 вершин, соединены каждая пятая вершина, как и в графе с 40 вершинами. В последнем рассмотренном графе со 100 вершинами соединены каждая пятая вершина. Во всех данных графах каждая вершина соединена с предыдущей.

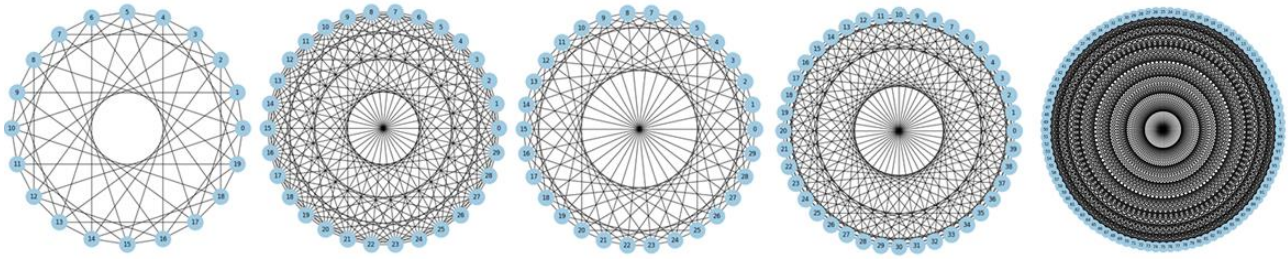

Рис. 4.1. Регулярные графы.

В данном случае минимальность по стягиванию занимает меньше времени, чем проверка на k-связность, потому что приведенные регулярные графы не являются минимальными по стягиванию. Из этого следует, что в определенный момент проверки происходит прерывание, так как данные графы не обладают этим свойством.

Далее были рассмотрены временные затраты на перебор графов. Так как количество графов растет экспоненциально, то было принято решение о введение потоков в текущую часть программы с перебором с целью повышения производительности. В таблицах, приведенных ниже, можно заметить, что оптимизация распараллеливанием снизила временные затраты по сравнению с предыдущей организацией структуры программы.

Таблица 4.3

Время перебора и поиска графов с распараллеливанием без рекурсии

	Количество	**Время**
N = 6, k = 4	15	318мс
N = 7, k = 5	105	2с
N = 8, k = 6	105	6с

Таблица 4.4

Время перебора и поиска графов без распараллеливания с/без рекурсии

	Количество	**Время (с)**	**Время (без)**
N = 6, k = 4	15	2с	345мс
N = 7, k = 5	105	3с	3с
N = 8, k = 6	105	14с	14с

При поиске минимальных и минимальных по стягиванию 4-связных графов на 6 вершинах алгоритм нашел и впоследствии визуализировал 15 графов, которые изображены на рис. 4.2.

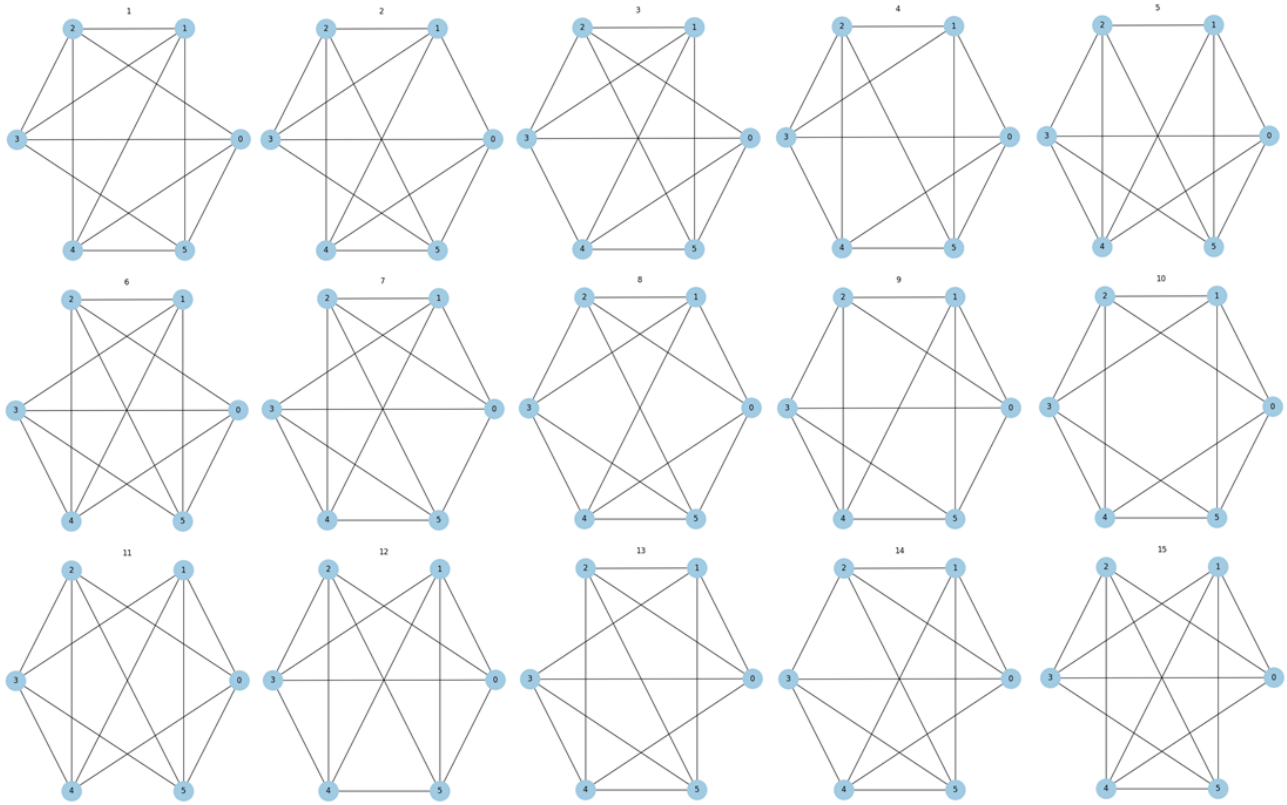

Рис. 4.2. Минимальные и минимальные по стягиванию 4-связные графы на 6 вершинах.

При нахождении графов, учитывается их повтор. То есть функция $compareMatrix$ сравнивает графы с нужными свойствами между собой и оставляет в итоге по одному экземпляру каждого графа.

Для проверки корректной работы программы были проведены эксперименты с минимальными и минимальными по стягиванию 5-связным, 6-связным и 7-связным графами из статьи С.А. Образцовой и А.В. Пастора [21]. Затраченное время на проверку таких графов приведено в табл. 4.5.

Таблица 4.5

Время работы проверок для графов из литературы

time / **N**	t_k	t_{min}	$t_{minContr}$	t
N = 17, k = 5	12мс	1мс	80мс	93мс
N = 15, k = 6	6мс	1мс	81мс	88мс
N = 19, k = 7	16мс	1мс	321мс	338мс

Результат проверок совпал с ожидаемым выводом в статье [21].

На рис. 4.3 приведена визуализация данных графов.

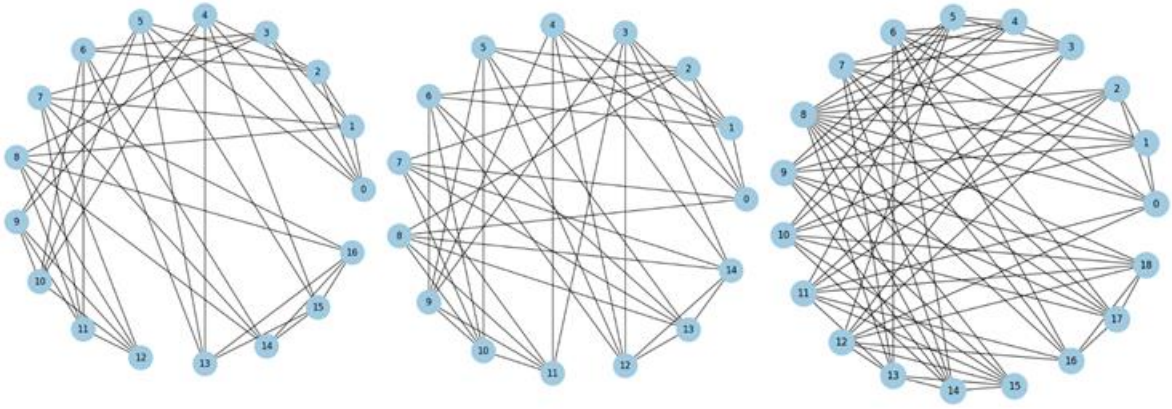

Рис. 4.3. Минимальные и минимальные по стягиванию 5-связный, 6-связный и 7-связный графы.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была разработана программа, которая совершает поиск минимальных и минимальных по стягиванию k-связных графов из полного (по умолчанию) или заданного графа. В качестве алгоритма проверки на k-связность были выбраны алгоритм Эвена и алгоритм Диница в качестве дополнения. Алгоритмы на проверки минимальности и минимальности по стягиванию были получены с помощью известных свойств минимальных и минимальных по стягиванию k-связных графов.

Помимо этого, было проанализировано несколько известных алгоритмов для проверки графа на k-связность, такие как алгоритмы Эвена, Диница, Форда-Фалкерсона, улучшенный алгоритм Галила. В результате изучения и анализа были выбраны наиболее подходящие алгоритмы для реализации программы. В дополнение были рассмотрены имеющиеся на сегодняшний день результаты в области изучения минимальных и минимальных по стягиванию k-связных графов.

После проведения ряда экспериментов были найдены минимальные и минимальные по стягиванию k-связные графы для малого значения N. Начальная реализация справлялась с данной задачей большое количество времени, поэтому были предприняты действия по улучшению кода и распараллеливанию на потоки процесса перебора графов при поиске.

Также были произведены замеры времени работы алгоритма проверки для различных графов с большим значением N: полные графы, регулярные графы. После ряда оптимизаций проверки для большого числа вершин, в частности это можно увидеть для полных графов, затраченное время сократилось. Но данные полные графы при этом не обладают рассматриваемыми нами свойствами, поэтому были рассмотрены регулярные графы. Такие графы точно обладают свойством k-связности, вследствие чего программа работала дольше.

В данной работе была разработана функция визуализации графов. Полученные графы, которые обладают нужными свойствами, были записаны в файл и переданы в программу визуализации. После чего программа отрисовки с помощью специального пакета NetworkX последовательно изображала каждый граф из указанного файла.

Полученные результаты свидетельствуют о том, что разработанная программа успешно осуществляет поиск минимальных и минимальных по стягиванию k-связных графов, проверку заданного графа на свойства k-связности, минимальности и минимальности по стягиванию и визуализации графов.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Kamalesh V. N. et al. On the design of fault tolerant k-connected network topologies //Int. J. Innov. Manag. Technol. – 2015. – Т. 6. – №. 5. – С. 339-342.

2. Latha S., Srivatsa S. K. Topological design of a K-connected communication network //WSEAS transactions on communications. – 2007. – Т. 6. – №. 4. – С. 657-662.

3. Карпов Д. В. Теория графов //СПб.: Санкт-Петербургское отделение Мат. института им. ВА Стеклова РАН. – 2017.

4. Menger K. Zur allgemeinen kurventheorie //Fundamenta Mathematicae. – 1927. – Т. 10. – №. 1. – С. 96-115.

5. Whitney H. Congruent graphs and the connectivity of graphs //Hassler Whitney Collected Papers. – Birkhäuser Boston, 1992. – С. 61-79.

6. Mader W. Ecken vom Gradn in minimalenn-fach zusammenhängenden Graphen //Archiv der Mathematik. – 1972. – Т. 23. – №. 1. – С. 219-224.

7. Mader W. Zur Struktur minimal n-fach zusammenhängender Graphen //Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg. – Springer-Verlag, 1979. – Т. 49. – №. 1. – С. 49-69.

8. Tutte W. T. A theory of 3-connected graphs //Indag. Math. – 1961. – Т. 23. – №. 441-455. – С.

9. Halin R. A theorem on n-connected graphs //Journal of Combinatorial Theory. – 1969. – Т. 7. – №. 2. – С. 150-154.

10. Fontet M. et al. Graphes 4-essentiels //C. R. Acad. Se. Paris, 287, serie A. – 1978. – С. 289–290.

11. Martinov N. A recursive characterization of the 4-connected graphs //Discrete mathematics. – 1990. – Т. 84. – №. 1. – С. 105-108.

12. Ando K., Kaneko A., Kawarabayashi K. Vertices of degree 5 in a contraction critically 5-connected graph //Graphs and Combinatorics. – 2005. – Т. 21. – №. 1. – С. 27-37.

13. Ando K., Chengfu Q. Some structural properties of minimally contraction-critically 5-connected graphs //Discrete mathematics. – 2011. – Т. 311. – №. 13. – С. 1084-1097.

14. Qin C., Yuan X., Su J. Some properties of contraction-critical 5-connected graphs //Discrete mathematics. – 2008. – Т. 308. – №. 23. – С. 5742-5756.

15. Ando K., Iwase T. The number of vertices of degree 5 in a contraction-critically 5-connected graph //Discrete mathematics. – 2011. – Т. 311. – №. 17. – С. 1925-1939.

16. Li M., Yuan X., Su J. The number of vertices of degree 7 in a contraction-critical 7-connected graph //Discrete mathematics. – 2008. – Т. 308. – №. 24. – С. 6262-6268.

17. Esfahanian A. H. Connectivity algorithms //Topics in structural graph theory. – Cambridge University Press, 2013. – С. 268-281.

18. Карпов Д. В., Пастор А. В. О структуре k-связного графа //Записки научных семинаров ПОМИ. – 2000. – Т. 266. – С. 76-106.

19. Образцова С. А. К-связные графы //Компьютерные инструменты в образовании. – 2010. – №. 1.

20. Карпов Д. В. Минимальные k-связные графы с минимальным числом вершин степени k //Записки научных семинаров ПОМИ. – 2014. – Т. 427. – №. 0. – С. 41-65.

21. Образцова С. А., Пастор А. В. О вершинах степени k минимальных и минимальных относительно стягивания k-связных графов: верхние оценки //Записки научных семинаров ПОМИ. – 2011. – Т. 391. – С. 198-210.

22. Образцова С. А. О локальной структуре 5 и 6-связных графов //Записки научных семинаров ПОМИ. – 2010. – Т. 381. – №. 0. – С. 88-96.

23. Пастор А. В. О вершинах степени 6 минимального и минимального по стягиванию 6-связного графа //Записки научных семинаров ПОМИ. – 2019. – Т. 488. – №. 0. – С. 143-167.

24. Even S. An algorithm for determining whether the connectivity of a graph is at least k //SIAM Journal on Computing. – 1975. – Т. 4. – №. 3. – С. 393-396.

25. Кормен Т. и др. Алгоритмы. Построение и анализ:[пер. с англ.]. – Издательский дом Вильямс, 2009.

26. Even S., Tarjan R. E. Network flow and testing graph connectivity //SIAM journal on computing. – 1975. – Т. 4. – №. 4. – С. 507-518.

27. Galil Z. Finding the vertex connectivity of graphs //SIAM Journal on Computing. – 1980. – Т. 9. – №. 1. – С. 197-199.

28. Girkar M., Sohoni M. On finding the vertex connectivity of graphs //Coordinated Science Laboratory Report no. UILU-ENG-87-2232, ACT-77. – 1987.

29. Schult D. A., Swart P. Exploring network structure, dynamics, and function using NetworkX //Proceedings of the 7th Python in science conferences (SciPy 2008). – 2008. – Т. 2008. – С. 11-16.

30. Диниц Е. А. Алгоритм решения задачи о максимальном потоке в сети со степенной оценкой //Доклады Академии наук. – Академия наук СССР, 1970. – Т. 194. – №. 4. – С. 754-757.