Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Саратовский национальный исследовательский государственный
университет имени Н. Г. Чернышевского»

На правах рукописи

Разумовский Пётр Владимирович

МИНИМАЛЬНЫЕ РАСШИРЕНИЯ ЦВЕТНЫХ ГРАФОВ

Специальность: 01.01.09 –
дискретная математика и математическая кибернетика

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
д.ф.-м.н., доц. Абросимов Михаил Борисович

Саратов 2022

Оглавление

Введение

Актуальность темы исследования и степень её разработанности. Влияние информационных технологий и технических систем становится все более значимым в разных важных сферах жизни общества: медицине, энергетике, транспортной связи, освоении космоса и многих других. Выход из строя хотя бы одного элемента системы, внедренной в данные сферы жизни, иногда приводит к катастрофическим последствиям. Поэтому использование вычислительных систем в критических областях формирует требование надежности. Надежность является комплексным свойством, включающим в себя такие понятия как долговечность, ремонтопригодность, сохраняемость, достоверность функционирования и, конечно же, отказоустойчивость. Говоря о построении надежной системы, чаще всего подразумевают именно свойство отказоустойчивости (fault tolerance).

Одна из первых работ, исследующая принципы построения отказоустойчивых систем, принадлежит Джону фон Нейману [101]. В ней рассматривается надежная система, собранная из ненадежных элементов. Говоря о повышении надежности самих элементов следует отметить, что, скорее всего, невозможно создать элемент идеальной надежности. Для каждого элемента системы принято использовать характеристику наработки на отказ. Время наработки элемента на отказ составляет то время, которое элемент проработает до выхода из строя. Так, например, ENIAC, одно из первых электронных вычислительных устройств, использовал около 18000 вакуумных ламп со средним временем наработки на отказ около 2500 часов. Среднее время наработки на отказ ENIAC составляло 7 минут [1, 56]. В современных компьютерах, конечно же, используются намного более надежные компоненты. В связи с этим выросла комплексность и сложность вычислительных устройств. Так, в 2020 году компания RIKEN объявила о запуске самого быстрого суперкомпьютера Фугаку

(Fugaku). 9 марта 2021 года [104] RIKEN объявила, что в этом компьютере установлено 158 976 процессоров Fujitsu A64FX [43, 100] с общим числом 7 299 072 ядер.

Исходя из этого, вопрос о надежности системы необходимо переформулировать в контексте способности системы продолжать работу при выходе надежного элемента из строя. В 1971 году А. Авиженис (A. Avižienis) [45] рассмотрел два подхода для повышения надёжности вычислительных систем: предотвращение ошибок и отказоустойчивость. Первое направление связано с уменьшением вероятности возникновения ошибки. Оно состоит в разработке высоконадёжных компонентов системы и не будет рассматриваться в данной работе. Во втором направлении используется введение в систему избыточных структур для придания ей свойств отказоустойчивости. В более поздних работах отказоустойчивость стала рассматриваться как одно из средств достижения гарантоспособности системы [47].

Авиженис [44] ввел также понятие *отказоустойчивости* как обеспечение системы способностью противостоять ошибке и возможностью продолжать работу в присутствии этой ошибки. Он рассмотрел два уровня отказоустойчивости:

1. *полная отказоустойчивость* – система продолжает работать в присутствии ошибок без существенной потери функциональных свойств;

2. *амортизация отказов* – система продолжает работать в присутствии ошибок с частичной деградацией функциональных возможностей.

В 1976 году John Hayes [79] предложил модель для исследования полной отказоустойчивости технических систем, основанную на графах. Некоторой системе Σ сопоставляется помеченный граф $G(\Sigma)$, вершины которого соответствуют элементам системы Σ, рёбра – связям между элементами, а метки указывают тип элементов. Если связи не являются симметричными, то можно рассматривать ориентированные графы. В данной работе будут рассматриваться неориентированные графы. Под *отказом* элемента системы Σ понимается удаление соответствующей ему вершины из графа системы $G(\Sigma)$ и всех связанных

с ней ребер. Система Σ^* называется *k-отказоустойчивой реализацией* системы Σ, если отказ любых k элементов системы Σ^* приводит к графу, в который можно вложить граф системы Σ с учетом меток вершин. Построение k-отказоустойчивой реализации системы Σ можно представить себе как введение в неё определенного числа избыточных элементов и связей. При этом предполагается, что в нормальном режиме работы избыточные элементы и связи *маскируются*, а в случае отказа происходит *реконфигурация* системы до исходной структуры [1, 79]. Есть и другая интерпретация отказа, при которой сам элемент перестаёт функционировать, но через него могут передаваться сигналы [28, 31]. На языке теории графов k-отказоустойчивую реализацию системы Σ мы будем называть вершинным k-расширением (В-kР) её графа $G(\Sigma)$.

Если в системе Σ встречается p различных типов элементов, то любая её k-отказоустойчивая реализация должна содержать не менее k дополнительных элементов каждого типа. Такого числа дополнительных элементов достаточно для построения k-отказоустойчивой реализации системы Σ. Добавим k элементов каждого типа, соединим их все между собой и с элементами системы Σ. Тогда любой отказавший элемент можно будет заменить одним из добавленных элементов соответствующего типа. Построенную k-отказоустойчивую реализацию будем называть *тривиальной.* Её можно использовать для оценки числа дополнительных вершин и рёбер произвольной k-отказоустойчивой реализации. Таким образом, известно минимально возможное число дополнительных элементов для построения k-отказоустойчивой реализации системы. Далее добавим условие минимальности числа дополнительных связей.

Если элементы технической системы однотипны, то метки элементов опускаются, и в качестве графа системы рассматривается граф без меток. В этом случае оптимальная k-отказоустойчивая реализация будет содержать в точности k дополнительных элементов. На практике такая ситуация встречается достаточно часто. Например, массивно-параллельные вычислительные системы состоят из однотипных узлов. Каждый узел состоит из одного или

нескольких процессоров, локальной памяти и некоторых других компонент [67].

В данной работе будут рассмотрены только помеченные графы, то есть системы с элементами i различных типов, где $i > 1$.

Для системы Σ ее k-отказоустойчивая реализация Σ^* называется *оптимальной*, если система Σ^* отличается от системы Σ на k элементов (каждого типа в случае помеченных графов), и среди всех k-отказоустойчивых реализаций с тем же числом элементов система Σ^* имеет наименьшее число связей. На языке теории графов оптимальную k-отказоустойчивую реализацию системы Σ мы будем называть минимальным вершинным k-расширением (MB-kP) её графа $G(\Sigma)$. Позднее F. Harary и J. Hayes [77] расширили модель на случай отказов связей. В данной работе мы будем рассматривать как отказы элементов, так и отказы связей между элементами.

В своей первой работе J. Hayes [79] предложил схемы построения минимальных вершинных 1-расширений для цепей и циклов, а также для частного случая помеченного дерева. Большое количество теоретических исследований посвящено аналитическому построению минимальных вершинных k-расширений для различных классов графов: И.А.К. Камил [9], Х.Х.К. Судани [15], П.П. Бондаренко [5], А.В. Киреева [32], М.Ф. Каравай [28–31], М.А. Кабанов [22], М.Б. Абросимов [1], С.Г. Курносова [33, 34], А.А. Долгов [19, 20], О.В. Моденова [11, 12], А.В. Гавриков [18], J. Hayes [39, 78, 79], A.A. Farrag [70] и ряд других работ [55, 59, 60, 80, 99].

Задача поиска минимального расширения для неориентированного помеченного графа является вычислительно сложной. А именно, задача проверки того, что граф является k-расширением заданного графа, является NP-полной [3]. Очевидно, поскольку граф является помеченным (цветным), вложение исходного графа следует проверять с условием сохранности цветов, то есть проверять графы на цветное вложение. В работах [1, 2] предлагается переборный алгоритм построения минимальных вершинных k-расширений графов. Его, конечно, можно модифицировать таким образом, чтобы проверялось

условие цветного вложения, однако это не отменяет факта, что описанный алгоритм сложен для распараллеливания и является неэффективным. Для генерации сложных комбинаторных структур, в том числе графов, хорошо показывают себя методы без непосредственной проверки на изоморфизм. В основе таких методов лежит каноническая форма структуры или её канонический код. Основная идея состоит в оставлении структуры только в том случае, если её форма является канонической. Необходимо назвать некоторые работы, пытающиеся раскрыть данную тему: И.А. Фараджев [42 ,69], С.А. Сухов [38], R.C. Read [61, 62, 97], B. McKay [91, 93, 94], G. Brinkmann [49–53], M. Meringer [95], R. Grund [75]. Алгоритмы генерации без проверки на изоморфизм хорошо подходят для распараллеливания.

На основании метода канонических представителей были разработаны несколько алгоритмов поиска минимальных расширений для непомеченных графов. В работе [9] предложен алгоритм для поиска минимальных вершинных расширений, а в работе [15] – минимальных реберных расширений. В данной работе предлагаются алгоритмы для цветных графов. Применение всех перечисленных алгоритмов ограничивается графами с небольшим числом вершин.

Аналитическое решение задачи поиска минимальных вершинных и реберных расширений позволяет найти схемы для цветных графов произвольного числа вершин и ребер, таким образом, решая вопрос для определенного класса графов.

Это определяет актуальность настоящей работы, а также соответствующие цели и задачи.

Цели и задачи диссертационной работы. Основные цели данной работы состоят в следующем:

1. Исследование алгоритмов для решения задачи генерации всех попарно-неизоморфных цветных графов без проверки на изоморфизм.

2. Исследование алгоритмов для решения задачи построения всех неизоморфных минимальных вершинных и реберных k-расширений для заданного цветного графа без проверки на изоморфизм.

3. Исследование минимальных вершинных k-расширений для цветных полных графов.

4. Исследование минимальных вершинных и реберных k-расширений для цветных звезд.

Основные задачи работы:

1. Разработка алгоритма построения всех попарно-неизоморфных цветных графов без проверки на изоморфизм.

2. Разработка алгоритма построения всех минимальных вершинных и реберных k-расширений для цветных графов без проверки на изоморфизм.

3. Получение схем для построения всех неизоморфных минимальных вершинных k-расширений для цветных полных графов.

4. Получение схем для построения всех неизоморфных минимальных вершинных и реберных k-расширений для цветных звезд.

Научная новизна работы заключается в следующем:

1. Разработаны новые алгоритмы построения всех попарно-неизоморфных цветных графов без непосредственной проверки на изоморфизм.

2. Разработаны новые алгоритмы построения всех неизоморфных минимальных вершинных и реберных k-расширений заданного цветного графа без непосредственной проверки на изоморфизм.

3. Найдены схемы построения минимальных вершинных k-расширений для цветных полных графов.

4. Найдены схемы построения минимальных вершинных и реберных k-расширений для цветных звезд.

Методы исследования. В работе используются методы дискретной математики и математической кибернетики. Используется аппарат теории графов, комбинаторики и теории алгоритмов.

Положения диссертации, выносимые на защиту:

1. Алгоритмы построения всех попарно-неизоморфных цветных реализаций заданного графа без непосредственной проверки на изоморфизм.

2. Алгоритмы построения всех неизоморфных минимальных вершинных и реберных k-расширений заданного цветного графа без непосредственной проверки на изоморфизм.

3. Схемы построения минимальных вершинных k-расширений для цветных полных графов.

4. Схемы построения минимальных вершинных и реберных k-расширений для цветных звезд.

Теоретическая и практическая значимость. Работа носит преимущественно теоретических характер и ее теоретическая значимость состоит в том, что удалось разработать алгоритмы построения всех цветных реализаций заданного графа, а также алгоритмы построения минимальных вершинных и реберных расширений цветных графов без проверки на изоморфизм. Для полных цветных графов и цветных звезд удалось найти полное аналитическое решение задачи построения их минимальных вершинных и реберных расширений.

Практическая значимость работы состоит в том, что на основе предложенных алгоритмов можно разрабатывать программные реализации, предназначенные для исследования минимальных расширений цветных графов, а с точки зрения приложений – изучать отказоустойчивые реализации технических систем с элементами разного типа. Автором были реализованы все алгоритмы в программном комплексе *ColorGraphExtensions*, который позволяет строить цветные графы и находить их минимальные вершинные и рёберные расширения.

Личный вклад соискателя. Все основные результаты, выносимые на защиту, получены автором лично. Научному руководителю принадлежат общее руководство, предложения по используемым методам, а также помощь в подготовке текста.

Структура и объем работы. Диссертационная работа состоит из введения, 4 глав, заключения, списка сокращений и условных обозначений, списка использованной литературы и приложений. Работа содержит 147 страниц, 52 рисунка, 44 таблицы, библиографический список из 104 наименований и 1 приложение.

Во введении обосновывается актуальность темы диссертационной работы, представлены обзор литературы по теме исследования, цели и задачи работы, научная новизна диссертации, теоретическая и практическая значимость работы, методы диссертационного исследования, основные результаты диссертации, структура работы, а также представлены степень достоверности результатов работ, апробации результатов работ и публикации по теме диссертации.

В первой главе приводятся некоторые понятия и обозначения из теории графов, предоставляющие теоретическую базу в области цветных графов и их изоморфизмов. Также приводятся известные и предлагаются новые алгоритмы, которые будут использоваться в диссертации для построения множества попарно-неизоморфных цветных реализаций для заданного непомеченного графа. Новые разработанные алгоритмы реализованы и включены в программный комплекс *ColorGraphExtensions.* После описания алгоритмов приводятся результаты различных вычислительных экспериментов запуска реализованных алгоритмов на различных классах графов. Данные были собраны в базу цветных графов следующих классов: цепи, циклы, полные графы и звезды.

Во второй главе приводятся некоторые понятия и обозначения из теории графов, предоставляющие теоретическую базу в области минимальных вершинных и реберных расширений. Предлагается уникальный способ кодирования цветных графов для решения задачи поиска минимальных расширений с применением техники канонических представителей. На основании данного кода представлен алгоритм для поиска всех неизоморфных минимальных рас-

ширений для заданного цветного графа. Реализация данного алгоритма включена в программный комплекс *ColorGraphExtensions*. Проведены вычислительные эксперименты на различных классах цветных графов, в том числе для цветных полных графов и цветных звезд. Для этих классов графов сформирована база минимальных вершинных и реберных расширений.

Третья глава целиком посвящена аналитическому решению задачи поиска минимальных вершинных расширений для цветных полных графов. Очевидно, что не существует реберных расширений полных графов, поэтому рассматриваются только минимальные вершинные расширения. Глава содержит рассуждения о схемах построения минимальных расширений, каждый шаг рассуждений обобщает предыдущие полученные и доказанные результаты. Полученные схемы согласуются с минимальными расширениями, полученными в ходе вычислительных экспериментов, и полностью решают вопрос о минимальных вершинных k-расширениях цветных полных графов.

В четвертой главе приводятся схемы построения минимальных расширений для звездных графов, и показано, что покрываются все возможные виды звездных графов с введенной на них функцией раскраски. Глава разбита на две части. *Первая часть главы 4* сосредоточена на поиске схем построения для минимальных реберных расширений цветных звезд. *Во второй части главы 4* приводятся теоремы с доказательствами со схемами построения минимальных вершинных расширений для цветных звезд. В данной главе полностью решен вопрос о минимальных расширениях цветных звезд.

Достоверности и апробация результатов исследования. Все полученные результаты снабжены строгими доказательствами и согласуются с результатами вычислительных экспериментов. Основные результаты работы докладывались и обсуждались на следующих семинарах и конференциях:

– научный семинар кафедры теоретических основ компьютерной безопасности и криптографии Саратовского национального исследовательского государственного университета имени Н.Г. Чернышевского (Саратов, 2017, 2018, 2019, 2020);

– научная конференция механико-математического факультета Саратовского национального исследовательского государственного университета имени Н.Г. Чернышевского «Актуальные проблемы математики и механики» (Саратов, 2021);

– VII Международная научная конференция «Компьютерные науки и информационные технологии» памяти А.М. Богомолова (Саратов, 2016);

– VIII Международная научная конференция «Компьютерные науки и информационные технологии» памяти А.М. Богомолова (Саратов, 2018);

– IX Международная научная конференция «Компьютерные науки и информационные технологии» памяти А.М. Богомолова (Саратов, 2021);

– всероссийская конференция «XVI Сибирская научная школа-семинар с международным участием «Компьютерная безопасность и криптография» – SIBECRYPT’17» (Томск, 2017);

– всероссийская конференция «XVIII Сибирская научная школа-семинар с международным участием «Компьютерная безопасность и криптография» – SIBECRYPT’19» (Томск, 2019);

– XX Международная конференция «Сибирская научная школа-семинар «Компьютерная безопасность и криптография» – SIBECRYPT’21» (Новосибирск, 2021);

– международная научная конференция студентов, аспирантов и молодых учёных «Ломоносов-2021» (Москва, 2021).

Все результаты диссертации, полученные автором, являются новыми и достоверными. Это подтверждается публикациями в рецензируемых научных изданиях. По теме диссертации имеется 3 публикации в изданиях из перечня рецензируемых научных изданий ВАК Минобрнауки РФ:

– Абросимов М.Б., Разумовский П.В. Минимальные расширения цветных звездных графов // International Journal of Open Information Technologies. – 2022. – Vol. 10, № 2. – P. 1–7.

– Разумовский П.В., Абросимов М.Б. Минимальные вершинные расширения цветных полных графов [Razumovsky P.V., Abrosimov M.B. The Minimal

Vertex Extensions for Colored Complete Graphs] // *Вестник ЮУрГУ. Серия «Математика. Механика. Физика»*. – 2021. – Т. 13, № 4. – С. 77–89.

– Абросимов М.Б., Разумовский П.В. О поиске минимальных вершинных расширений цветного неориентированного графа // *Известия высших учебных заведений. Поволжский регион. Физико-математические науки.* – 2021. – № 4 (60). – С. 106–117.

Следующие публикации по теме диссертации были опубликованы в научных изданиях, которые входят в базы цитирования Web of Science и Scopus:

– Разумовский П.В., Абросимов М.Б. Построение цветных графов без проверки на изоморфизм // *Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика*. 2021. Т. 21, вып. 2. С. 267–277.

– Razumovsky P.V. The search for minimal edge 1-extension of an undirected colored graph [Разумовский П. В. О поиске минимальных реберных 1-расширений неориентированного цветного графа] // *Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика*. 2021. Т. 21, вып. 3. С. 400–407.

Программный комплекс *ColorGraphExtensions* был зарегистрирован как объект интеллектуальной собственности:

– Разумовский П.В., Абросимов М.Б. *ColorGraphExtensions* // Свидетельство о государственной регистрации программы для ЭВМ № 2021662022, выданное Роспатентом. Зарегистрировано в Реестре программ для ЭВМ 20.06.2021.

Также по теме диссертации автором публиковались следующие работы:

– Абросимов М.Б., Разумовский П.В. Генерация неизоморфных вершинных k-раскрасок графа // *Компьютерные науки и информационные технологии: материалы Междунар. науч. конф.* – Саратов: центр «Наука», 2016. – С. 13–15.

– Абросимов М.Б., Разумовский П.В. О генерации неизоморфных k-раскрасок методом МакКея // *Компьютерные науки и информационные технологии: материалы Междунар. науч. конф.* – Саратов: центр «Наука», 2018. – С. 318–320.

– Абросимов М.Б., Разумовский П.В. О минимальных расширениях неизоморфных цветных звездных графов // *Компьютерные науки и информационные технологии: материалы Междунар. науч. конф.* – Саратов: центр «Наука», 2021. – С. 5–8.

– Абросимов М.Б., Разумовский П.В. О генерации неизоморфных вершинных k-раскрасок // *ПДМ. Приложение*. – 2017. – № 10. – С. 136–138.

– Абросимов М.Б., Разумовский П.В. О генерации неизоморфных раскрасок методом Рида-Фараджева // *ПДМ. Приложение.* – 2019. – № 12. – С. 173–176.

– Разумовский П.В. О минимальных вершинных 1-расширениях двухцветных полных графов // *Материалы Международного молодежного научного форума «ЛОМОНОСОВ-2021»* [Электронный ресурс] – 2021. – URL: https://lomonosov-msu.ru/archive/Lomonosov_2021/data/22112/124513_uid563707_ report.pdf (дата обращения 25.11.2021).

Автор выражает глубокую признательность своему научному руководителю доктору физико-математических наук Михаилу Борисовичу Абросимову за предложенную тему исследования и поставленные задачи, за интенсивную плодотворную многолетнюю работу, а также за всестороннюю помощь и поддержку при подготовке диссертационной работы.

Глава 1. Цветные реализации заданных графов

В первой главе определяются необходимые понятия и обозначения теории графов, которые будут использоваться на протяжении всей диссертации. Исследование сосредоточено на поиске решения задачи минимальных расширений цветных графов, однако на данный момент не существует достаточных программных решений и эффективных алгоритмов для построения множеств неизоморфных цветных графов. В данной главе ставится и решается задача построения всех неизоморфных цветных графов. Для этого предлагается эффективный алгоритм. На основании разработанного алгоритма создан программный комплекс, который позволяет получить все неизоморфные цветные реализации для заданного графа. Проведены вычислительные эксперименты для различных классов графов. В дальнейшем эти результаты позволят подтвердить аналитические рассуждения о минимальных расширениях цветных графов.

1.1 Основные определения теории графов

Введем необходимые основные определения из теории графов. Терминология приведена в соответствии с [17] и [39].

Определение 1.1.1. Пара $G = (V, \alpha)$ называется *неориентированным графом* (везде далее просто *графом*), где $V = \{1, \dots, n\}$ – множество вершин, а α – симметричное и антирефлексивное отношение на множестве вершин V.

Определенные таким образом графы иногда называют *простыми*, то есть не содержащими петель и кратных ребер.

Определение 1.1.2. Если $(u, v) \in \alpha$, где $u, v \in V$, то говорят, что вершины u и v *смежны* и соединены *ребром* (u, v).

В терминологии неориентированных графов $(u, v) \in \alpha$ и $(v, u) \in \alpha$ это одно и то же ребро, которое обозначают $\{u, v\}$.

Определение 1.1.3. Ребро $\{u, v\} \in \alpha$ называют *инцидентным* каждой из вершин u и v.

Определение 1.1.4. Степенью $d(v)$ вершины v называют количество вершин в G, смежных с v.

Определение 1.1.5. Вектор, составленный из степеней вершин графа G в порядке невозрастания, называют *вектором степеней.*

Количество вершин в графе $G = (V, \alpha)$ принято обозначать $n = |V|$, а количество ребер $m = |\alpha|$. В дальнейшем в работе будет использоваться именно это обозначение для количества вершин и ребер.

Хотя в данной работе будут рассматриваться *неориентированные графы*, однако, многие рассуждения могут быть перенесены и на ориентированные графы.

Определение 1.1.6. Граф, любые две вершины которого смежны, называется *полным* и обозначается K_n. Таким образом, $K_n = (V, V \times V - \Delta)$, где Δ является тождественным отношением на множестве вершин.

Определение 1.1.7. Граф без ребер, то есть с $|\alpha| = 0$, называется *вполне несвязным* и обозначается O_n.

Определение 1.1.8. Однородным, или *регулярным, n-вершинным графом порядка p* называют граф, в котором все степени вершин равны p. Подобный граф обозначается $R_{n,p}$.

Определение 1.1.9. Граф называется *двудольным*, если множество вершин V может быть разбито на два подмножества вершин V_1 и V_2 такие, что концы любого ребра графа принадлежат разным подмножествам.

Определение 1.1.10. Если двудольный граф содержит все ребра, соединяющие вершины из множеств V_1 и V_2, то он называется полным двудольным графом и обозначается $K_{m,n}$, где m и n – число вершин в множествах V_1 и V_2.

Определение 1.1.11. Граф $S = (V, \alpha)$, где $|V| = n$, $|\alpha| = m$ называется *звездой*, или *звездным графом*, если это граф вида $K_{1,n}$.

В дальнейшем корневую вершину будем называть *центром звезды* и обозначать c, $c \in V$.

Определение 1.1.12. Цепью, или *цепным графом*, называется граф $G = (V, \alpha)$, в котором $V = \{v_1, v_2, \dots, v_n\}$ и $\alpha = \{(v_i, v_j): |i - j| = 1\}$.

Определение 1.1.13. Граф $G = (V, \alpha)$ называют *циклом*, или *циклическим графом*, когда при множестве вершин $V = \{v_1, v_2, \dots, v_n\}$ множество ребер равно $\alpha = \{(v_i, v_j): |i - j| = 1\} \cup \{(v_1, v_n), (v_n, v_1)\}$.

Обычно считают, что циклы определены для числа вершин $n \geq 3$.

Определение 1.1.14. Соединением двух графов $G_1 = (V_1, \alpha_1)$ и $G_2 = (V_2, \alpha_2)$, не имеющих общих вершин, называют граф

$$G_1 + G_2 \coloneqq (V_1 \cup V_2, \alpha_1 \cup \alpha_2 \cup V_1 \times V_2 \cup V_2 \times V_1).$$

Определение 1.1.15. Объединением двух графов $G_1 = (V_1, \alpha_1)$ и $G_2 = (V_2, \alpha_2)$ называется граф $G_1 \cup G_2 = (V_1 \cup V_2, \alpha_1 \cup \alpha_2)$. Далее принимается правило, что при этом множества V_1 и V_2 не пересекаются.

Определение 1.1.16. Вложением графа $G_1 = (V_1, \alpha_1)$ в граф $G_2 = (V_2, \alpha_2)$ называется такое взаимно однозначное отображение $f: V_1 \to V_2$, что для любых двух вершин $u, v \in V_1$ выполняется следующее условие:

$$(u, v) \in \alpha_1 \Rightarrow (f(u), f(v)) \in \alpha_2$$

При этом говорят, что граф G_2 *допускает вложение* графа G_1.

Определение 1.1.17. Будем считать, что каждая вершина *достижима* из самой себя. Тогда отношение достижимости является отношением эквивалентности на множестве вершин графа. Классы этого отношения называются *компонентами связности* графа.

Определение 1.1.18. Граф с универсальным отношением достижимости называется *связным.* Другими словами, связный граф – это граф с единственной компонентой связности.

Определение 1.1.19. Два ребра $\{u_1, v_1\}$ и $\{u_2, v_2\}$, $\{u_1, v_1\} \neq \{u_2, v_2\}$, называются *смежными*, если $u_1 = u_2$ или $v_1 = v_2$.

Определение 1.1.20. Пусть S – множество, а $F = \{S_1, \dots, S_p\}$ – семейство его различных непустых подмножеств, объединение которых дает S. *Графом пересечений* $\Omega(F)$ семейства F называют такой граф, который определяется множеством $V(\Omega(F)) = F$ и условием: «S_i и S_j смежны тогда и только тогда, когда $i \neq j$ и $S_i \cap S_j \neq \emptyset$».

Определение 1.1.21. Рассмотрим множество α всех ребер графа $G = (V, \alpha)$ как семейство двухвершинных подмножеств множества вершин $V(G)$. Тогда *реберным графом* $L(G)$ графа G называется граф пересечений $\Omega(\alpha)$.

Исходя из данного определения, вершинами графа $L(G)$ являются ребра графа G, а две вершины из $L(G)$ смежны тогда и только тогда, когда смежны соответствующие им ребра графа G.

1.2 Изоморфизмы и автоморфизмы в теории графов

Теперь определим отношение изоморфизма и автоморфизма в теории графов. Термины будут использоваться в соответствии с [1] и [94].

Определение 1.2.1. Два графа $G_1 = (V_1, \alpha_1)$ и $G_2 = (V_2, \alpha_2)$ называются *изоморфными*, если можно установить взаимно однозначное соответствие $f: V_1 \to V_2$, сохраняющее отношение смежности между каждой вершиной u, v:

$$(u, v) \in \alpha_1 \Leftrightarrow (f(u), f(v)) \in \alpha_2$$

Изоморфизм двух графов обозначается $G_1 \cong G_2$.

Определение 1.2.2. Изоморфное отображение графа на себя называется *автоморфизмом*.

Из этого следует, что тождественное отображение $\Delta: V \to V$ является автоморфизмом для любого графа. Автоморфизмы графа образуют группу.

Определение 1.2.3. Две вершины u и v называются *подобными*, если существует автоморфизм f такой, что $f(u) = v$.

Определение 1.2.4. Множество подобных вершин графа называется его *орбитой.*

Если граф $G_1 = (V_1, \alpha_1)$ изоморфен графу $G_2 = (V_1, \alpha_1)$, то их реберные графы тоже изоморфны: $L(G_1) \cong L(G_2)$. Обратное утверждение справедливо почти всегда. В 1932 году Хасслер Уитни доказал [103], что графы G_1 и G_2, у которых $L(G_1) \cong L(G_2)$ изоморфны всегда, кроме случая, когда один из них есть K_3, а другой $K_{1,3}$. Данные графы представлены на рисунке 1.2.1.

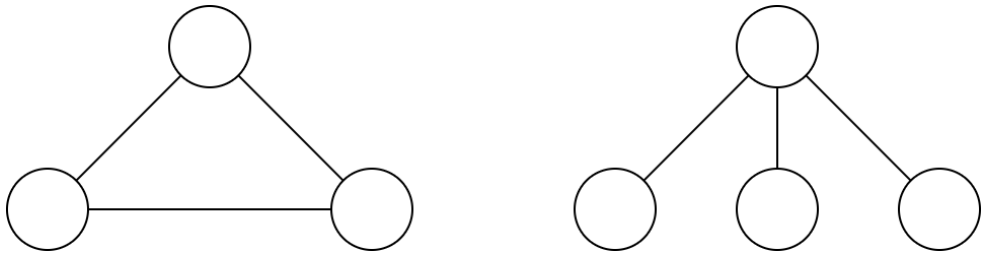

Рисунок 1.2.1 – Графы K_3 и $K_{1,3}$ соответственно.

Определение 1.2.6. Инвариантом графа G называется набор его характеристик, одинаковый для всех изоморфных ему графов.

Инвариантами графа являются, например, число вершин графа, количество дуг или рёбер.

Определение 1.2.7. Полным инвариантом графа называется такой инвариант, который определяет граф однозначно с точностью до изоморфизма.

Одним из известных числовых инвариантов является максимальный (минимальный) матричный код графа [89].

Определение 1.2.8. Матрица отношения смежности графа называется его *матрицей смежности.*

Пусть $G = (V, \alpha)$ – n-вершинный граф. Тогда его матрица смежности $A = M(\alpha)$ имеет размерность $n \times n$, а на позиции (i, j) стоит 1, если есть дуга (v_i, v_j), и 0 в противном случае:

$$A = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & & x_{2,n} \\ \vdots & & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{pmatrix}$$

Если выписать элементы матрицы смежности по строкам, то получится двоичное число, *код графа*:

$$x_{1,1}, x_{1,2}, \dots, x_{1,n}, x_{2,1}, x_{2,2}, \dots, x_{2,n}, \dots, x_{n,1}, x_{n,2}, \dots, x_{n,n}$$

Само по себе это число не является инвариантом, так как матрицы смежности у изоморфных графов могут различаться, однако если среди всех изоморфных графов выбрать матрицу смежности с максимальным (*максимальный код*) или минимальным (*минимальный код*) значением кода, то получится инвариант, причем полный. Очевидно, что для n-вершинного графа размер кода будет n^2 бит. Для некоторых классов графов, например, для неориентированных графов, достаточно хранить только часть матрицы смежности, расположенную над главной диагональю. В этом случае размер кода будет составлять $\frac{n(n-1)}{2}$ бит.

1.3 Цветные графы и их изоморфизм в теории графов

Поскольку в данной работе преимущественно рассматриваются цветные графы, необходимо дать необходимые термины и переложить интересующие части теории графов на случай цветных графов. Все понятия даны в соответствии с [1] и [96].

Определение 1.3.1. Граф, вершинам которого приписаны метки, называется *помеченным.*

Определение 1.3.2. Пусть $G = (V, \alpha)$ – граф, и $i \in \mathbb{N}$. Функция вида $f: V \to \{1, \dots, i\}$ называется *вершинной i-раскраской* графа G, а $f(v), v \in V$ называется *цветом* вершины v. При этом граф называется *графом с цветными вершинами*, или *цветным графом*. Для таких графов вводится следующее обозначение: $G = (V, \alpha, f)$.

В дальнейшем множество цветов, сопоставленных графу G будем обозначать $F = \{1, \dots, i\}$.

Очевидно, что помеченный граф и цветной граф – это два различных обозначения одного и того же вида графа.

Определение 1.3.3. Пусть существует граф $G = (V, \alpha)$ и $i \in \mathbb{N}$. Функция вида $\psi: \alpha \to \{1, \dots, i\}$ называется *реберной i-раскраской* графа G, а $\psi(\{u, v\}), \{u, v\} \in \alpha$ – цветом ребра $\{u, v\}$. При этом граф называется *графом с цветными ребрами* и обозначается $G = (V, \alpha, \psi)$.

Везде далее *вершинной раскраской* графа, или просто *раскраской* графа, будем называть вектор $\big(f(v_1), f(v_2), \dots, f(v_n)\big)$ – результат применения функции f к каждому элементу множества $V = \{v_1, v_2, \dots, v_n\}$.

Аналогично *реберной раскраской* графа будем называть результат применения функции ψ к каждому элементу множества α.

Определение 1.3.4. Изоморфизмом цветных графов $G_1 = (V_1, \alpha_1, f_1)$ и $G_2 = (V_2, \alpha_2, f_2)$ называется изоморфизм графов $G_1 = (V_1, \alpha_1)$ и $G_2 = (V_2, \alpha_2)$, сохраняющий цвета. Это биекция $\varphi: V_1 \to V_2$, при которой выполняются два условия:

1. $\forall\, u, v \in V_1: (u, v) \in \alpha_1 \Leftrightarrow \big(\varphi(u), \varphi(v)\big) \in \alpha_2$;
2. $\forall\, v \in V_1: f_1(v) = f_1\big(\varphi(v)\big)$.

Изоморфизм цветных графов также называется *цветным изоморфизмом.* Аналогично вводится понятие *изоморфизма графов с цветными ребрами.*

Также часто применяется понятие, что графы $G_1 = (V_1, \alpha_1, f_1)$ и $G_2 = (V_2, \alpha_2, f_2)$ изоморфны *с учетом цветов.*

Определение 1.3.5. Цветной автоморфизм графа – цветной изоморфизм графа на себя.

1.4 Задача о поиске неизоморфных цветных графов

Одной из задач теории графов является проблема нахождения цветных графов (и графов с цветными ребрами). Раскраски заданных структур рассматривались еще с конца 19-го века. Например, проблема четырех раскрасок была сформулирована и опубликована в 1879 году английским математиком Артуром Кэли [57]. Он рассматривал данную проблему раскраски топологических карт.

Дальнейшим развитием было переложение задач раскраски подобного рода на элементы теории графов. Так, в 1949 году А. Зыков в работе [21] предложил прочтение проблемы четырех раскрасок как задачу о поиске помеченного графа (в контексте этой работы – цветного графа), который удовлетворял бы условиям задачи.

В виде алгоритмической задачи проблема раскраски графов стала активно исследоваться в 1970-х годах. Определение числа, при котором любые две смежные вершины имеют разный цвет, вошло в так называемую «21 NP-полную задачу Карпа» [83]. В том или ином виде задача раскраски графа фигурировала в работах [63], [48] и других подобных. Спустя несколько лет было найдено и практическое применение данной задачи. В работе [58] предлагается использование раскраски графа для распределения регистров в компиляторах. В ней рассматривалось применение техник генерации раскрасок графа

для глобальной аллокации регистров в экспериментальном PL/I оптимизирующем компиляторе.

Обычно генерация раскрасок заданного графа рассматривается в контексте поиска неизоморфных цветных графов, поскольку изоморфные друг другу графы не представляют особого интереса. Генерация всех неизоморфных цветных графов заданного количества вершин и множества ребер α ровно в i цветов не имеет эффективного решения. В работе [73] доказано, что задача, состоящая в том, чтобы узнать, допускает ли граф раскраску ровно в i цветов, является NP-полной кроме случаев $i = 1$ и $i = 2$.

1.5 Базовый алгоритм генерации неизоморфных цветных графов

Задачу генерации неизоморфных раскрасок можно решить следующим способом. Для заданного n-вершинного графа $G = (V, \alpha)$ применяется алгоритм полного перебора: каждому элементу множества $F = \{1, \dots i\}$ ставятся в соответствие элементы множества V. Таким образом, общее количество полученных графов будет равно i^n. Среди данного числа будут встречаться изоморфные графы. Поскольку задача генерации подразумевает получение неизоморфных цветных графов, из полученного множества мощности k^n необходимо выделить по одному представителю от всех изоморфных графов. Это можно выполнить точным способом исключения изоморфных графов. Данный алгоритм был описан в [13].

Алгоритм 1.5.1. Исключение изоморфных графов прямой проверкой

Вход: $S = \{G_1, \dots, G_n\}$.

1.1 $I \coloneqq \emptyset, j \coloneqq 1$.

1.2 $j = j + 1$.

1.3 Если $j > n$, то завершить алгоритм, где I будет списком попарно неизоморфных графов.

1.4 $\nexists H \in I: H \cong G_j$, то добавить G_j в I: $I \leftarrow G_j$.

1.5 Перейти к шагу 2.

Количество проверок варьируется от поданного на вход множества S. Если все n графов из данного множества являются изоморфными друг другу, то количество проверок будет равным $n - 1$. Если же все n графов попарно неизоморфны, то количество проверок увеличится до $\frac{n(n-1)}{2}$.

В целом количество проверок графов в данном алгоритме имеет вычислительную сложность $O(n^2)$.

Отдельной проблемой является сама проверка двух графов на изоморфизм. Задача распознавания изоморфизма графов относится к классу NP [72], но неизвестно, является она NP-полной или принадлежит классу полиномиально разрешимых. Хотя и существуют достаточно быстрые алгоритмы [94, 98], в том числе реализованные в программных комплексах, например, *Bliss* [81, 82], *conauto* [86, 87], *saucy* [64], *nauty* [88, 93] и прочие, задача проверки двух графов на существование изоморфизма между ними все еще остается вычислительно сложной. Необходимо заметить, что многие из приведенных алгоритмов позволяют строить каноническую форму графа, что будет рассмотрено далее в работе.

На практике алгоритм полного перебора является крайне неэффективным и упирается в пределы технических систем, на которых был запущен поиск для графов с некоторой конфигурацией. Было обнаружено, что для ускорения технических реализаций можно пользоваться разными способами оптимизации перебора путем уменьшения общего числа перебираемых вариантов. Подобный подход принято называть *перебором с отсечением*.

Одним из наиболее известных решений задачи поиска неизоморфных графов путем перебора с отсечениями является техника «*isomorphism rejection*», или техника *отсечения изоморфизмов.* Хороший обзор можно найти в работе [50].

В данном разделе представлены реализации двух наиболее популярных методов для случая цветных графов. Первый из них – это метод Брендана Мак-Кея [91], предложенный в 1998 году, а второй – метод Рида-Фараджева [50]. Оба этих подхода оперируют с канонической нумерацией графа, однако каждый из них отличается подходом к отсечению неподходящих вариантов цветных графов. Рассмотрим далее метод канонической нумерации, или канонических представителей, а также каждую из этих техник подробнее в контексте генерации неизоморфных цветных графов без проверки на изоморфизм.

1.6 Метод канонических представителей

Метод канонических представителей основывается на следующих общих принципах [50]:

1. Определяется способ кодирования объектов.

2. Среди всех кодов изоморфных объектов выбирается канонический код.

3. Порождаются все возможные уникальные структуры вместе с их кодами.

4. Порожденная структура принимается, если её код канонический, в противном случае – исключается.

Для примера работы данной техники рассмотрим *матричный код* графа. Это двоичное число, которое получается путем выписывания друг за другом строк или столбцов матрицы смежности графа. В случае неориентированных графов нет смысла выписывать элементы всех строк, можно ограничиться выписыванием элементов выше или ниже главной диагонали, поскольку неориентированный граф может быть однозначно восстановлен по верхнему или нижнему треугольнику матрицы смежности. В качестве канонического кода выберем *максимальный матричный код*. Из введенных определений следует, что максимальный матричный код является полным инвариантом, а значит, является однозначным способом кодирования объектов. Тогда получаем следующее утверждение:

Допустим, существует $c(G)$ *– некоторый матричный код графа, и* $c_{max}(G)$ *– максимальный код графа, изоморфного графу* G*. Если* $c(G) = c_{max}(G)$*, то граф* G *является каноническим представителем.*

Одними из первых задачу приведения графа к каноническому виду исследовали В.Л. Арлазаров и И.А. Фараджев в работе [16]. Более полную историю исследований канонических представителей в качестве подходящего метода для решения задачи отсечения изоморфизмов и ускорения генерации можно найти в работе [76].

Указанные методы МакКея и Рида-Фараджева основываются именно на данных принципах построения канонических представителей, однако различаются в подходе к порождению структур. Тем не менее, каждый из них является эффективным способом уменьшить общее число перебираемых структур.

1.7 Метод МакКея отсечения изоморфизмов

Брендан МакКей в своих работах [88, 89, 91, 93, 94] рассматривал технику канонических представителей с точки зрения итерационной генерации структур от меньших к большим. Он предложил рассматривать генерацию от родителя к потомку. При этом одна структура может быть потомком нескольких родителей.

Общий принцип метода МакКея заключается в следующем:

1. Для каждой структуры определяется уникальный родитель и способ, которым данная структура должна быть получена из него. Эта операция дает представление о каноническом представителе.
2. Структура принимается, только если она генерируется выбранным способом из канонического родителя.
3. Из каждого родителя все неизоморфные потомки должны генерироваться по одному разу.

При выполнении всех условий в результате получается множество неизоморфных структур.

Одной из важнейших операций в алгоритме МакКея вычисления неизоморфных структур является *уточнение разбиения графа*.

Определение 1.7.1. Разбиением графа называют представление множества вершин в виде упорядоченного набора его попарно непересекающихся непустых подмножеств. Разбиение графа обозначают $\left[v_{i_1}, v_{i_2}, \dots, v_{i_k} \middle| v_{i_{k+1}}, \dots, v_{i_{k+j}} \middle| v_{i_{k+j+1}}, \dots, v_{i_m}\right]$.

Например, разбиение графа может быть таким: $[v_1, v_2|v_3, v_4, v_5|v_6]$.

Так, например, функция раскраски графа определяет разбиение его вершин на цветовые классы.

Определение 1.7.2. Ячейкой разбиения графа называют подмножество разбиения и обозначают $[v_1, v_2, \dots, v_r]$.

Пример разбиения содержит три ячейки: $[v_1, v_2]$, $[v_3, v_4, v_5]$ и $[v_6]$.

Определение 1.7.3. Если каждая ячейка разбиения π_1 является подмножеством какой-либо ячейки разбиения π_2, то говорят, что π_1 является *подразбиением* π_2, а π_2 – надразбиением π_1 соответственно.

Здесь подразумевается, что при подразбиении сохраняется порядок ячеек. Например, $[1\ 2 \mid 3 \mid 4\ 5]$ является подразбиением $[1\ 2\ 3 \mid 4\ 5]$ и $[1\ 2 \mid 3\ 4\ 5]$, но не $[3 \mid 1\ 2\ 4\ 5]$.

Определение 1.7.4. Разбиение графа, каждая ячейка которого состоит из одного элемента, называют *дискретным.* Получается, что некоторое дискретное разбиение графа образует *перестановку* вершин.

Для примера – разбиение $[3 \mid 1 \mid 4 \mid 5 \mid 2]$ образует перестановку $(3, 1, 4, 5, 2)$.

Определение 1.7.5. Говорят, что разбиение графа *элементарное*, если оно состоит из одной ячейки.

Операция уточнения разбиения в алгоритме МакКея [91] базируется на поиске *справедливого подразбиения* (equitable partition). В работе [35] данная операция подробно рассмотрена со всеми особенностями, облегчающими понимание итоговой реализации данного алгоритма в виде программного комплекса *nauty* [92].

В данном смысле *справедливым подразбиением* называют разбиение π, для которого выполняется следующее условие: для любых двух ячеек $W_1, W_2 \in \pi$ и для любых двух вершин из первой ячейки $v_1, v_2 \in W_1$ выполняется равенство $d(v_1, W_2) = d(v_2, W_2)$. Функция d здесь является количеством вершин из ячейки W, смежных с вершиной v:

$$d(v, W) = |\{u: \forall\, u \in W(v, u) \in \alpha\}|.$$

Очевидно, что любое дискретное разбиение является справедливым.

Смысл работы алгоритма МакКея с разбиениями заключается в переборе и анализе разбиений. Первое встреченное дискретное разбиение сохраняется как ξ. Далее каждое последующее дискретное разбиение π сравнивается с разбиением ξ: если перестановка, образованная разбиением $\xi^{-1} \circ \pi$ является

автоморфизмом графа – она записывается в группу автоморфизмов. Здесь «∘» является операцией композиции π и перестановки, обратной ξ.

Использование автоморфизмов позволяет существенно сократить перебор при нахождении канонической перестановки. Поэтому в *nauty* и производных алгоритмах обязательно осуществляется нахождение группы автоморфизмов. В процессе построения группы автоморфизмов для заданного графа производятся четыре операции по сокращению перебора.

Первое сокращение позволяет подняться к общему родителю перестановок π и ξ при нахождении автоморфизма $\gamma = \xi^{-1} \circ \pi$, поскольку поддерево, в котором находится γ, изоморфно поддереву, в котором находится разбиение ξ.

Второе сокращение связано с нахождением «наилучшей» перестановки. Сравнение, какая из перестановок «лучше», определяется некоторой функцией, задающей транзитивное отношение, результат которой не должен зависеть от первоначальной нумерации графа. В качестве примера такой функции можно брать исходный граф G и перестановки π_1 и π_2, строить по ним графы G_{π_1} и G_{π_2}, вычислять коды их матрицы смежности и сравнивать лексикографически. Перестановка, дающая при сравнении наибольший код матрицы, будет считаться «лучше» второй.

Итак, обозначим «наилучшую» перестановку за ρ. Эта перестановка и будет хранить в себе каноническую нумерацию графа на момент завершения построения группы автоморфизмов. При обнаружении перестановки ξ, она копируется в ρ. Далее, при обнаружении каждой новой перестановки π, она сравнивается с ρ, и если π «лучше» ρ, то $\rho := \pi$. Если же они одинаковы, то $\rho^{-1} \circ \pi$ является автоморфизмом, а, значит, можно воспользоваться первым сокращением.

Третье сокращение работает с орбитами. Изначально каждая вершина принадлежит собственной орбите. По мере обнаружения новых автоморфизмов орбиты автоморфных вершин объединяются, а их количество – сокраща-

ется. Таким образом, нет необходимости производить разбиение, отделяя разные вершины в одной орбите, поскольку все перестановки данного поддерева будут изоморфны обнаруженным ранее перестановкам, и все найденные автоморфизмы будут повторять предыдущие.

Четвертое сокращение перебора позволяет обрабатывать такие разбиения, которые не могут быть обработаны предыдущим сокращением (например, разбиения, не являющиеся предками ξ). Для каждого автоморфизма γ хранятся два множества: $fix(\gamma)$ и $mcr(\gamma)$. Здесь $fix(\gamma)$ – множество вершин, которые остаются на своих местах при действии автоморфизма. А $mcr(\gamma)$ – это множество вершин, которые являются минимальными в каждой из орбит, составленных по автоморфизму γ. Допустим, существует множество V_i, элементы которого перебираются для продолжения обхода разбиений из некоторого потомка глубины l. В процессе пути до этого потомка l вершин уже были «зафиксированы». Обозначим множество этих вершин $fixed$. МакКей доказал в своей работе [91], что если $fixed \subset fix(\gamma)$, то можно взять за V_i множество $V_i \cap mcr(\gamma)$ без ущерба для алгоритма.

1.8 Метод Рида-Фараджева отсечения изоморфизмов

Метод Рида-Фараджева так же, как и предыдущий рассмотренный метод, является развитием идеи использования канонических представителей [50, 69, 97].

Идея метода Рида-Фараджева заключается в том, чтобы исключать неканонические структуры как можно раньше. Структура строится в несколько уровней приближения. На уровнях приближения могут происходить соответ-

ствующие данному уровню проверки – может ли потомок данного приближения быть каноническим. У итоговой структуры происходит проверка канонич-ности. В итоге достигается основная цель данного метода – на раннем этапе уменьшается количество итоговых структур, которые нужно проверять на каноничность.

Для примера работы метода Рида-Фараджева приведём следующую теорему.

Теорема 1.8.1. Пусть граф H отличается от графа G на $m > 0$ рёбер, и код графа H $\mathrm{c}(H) = x_1x_2\ldots x_k\,\mathbf{1}\,0\,\ldots 0$ является каноническим, тогда код $c' = x_1x_2\ldots x_k\,\mathbf{0}\,0\ldots0$, получающийся из $\mathrm{c}(H)$ заменой самой правой единицы на ноль, также будет каноническим.

Доказательство данной теоремы приведено в работе [25].

Из теоремы следует, что все n-вершинные графы с максимальными матричными кодами образуют дерево, которое является поддеревом дерева всех графов. Заметим, что любой n-вершинный граф может быть получен добавлением рёбер к графу O_n, то есть такое дерево уже было рассмотрено раньше. Если код какого-либо графа не является максимальным, то ни один из его дочерних элементов не является максимальным. То есть, всё поддерево с корнем в данной вершине можно не проверять – они не будут иметь максимальные матричные коды. В итоге для построения всех графов с максимальными матричными кодами нужно будет совершить меньше проверок.

1.9 Алгоритм генерации неизоморфных цветных графов методом МакКея

Алгоритмизируем генерацию неизоморфных раскрасок методом МакКея. В ходе алгоритма будет использоваться операция вычисления орбит графа. Данная операция может быть реализована по алгоритму МакКея, описанного в [91], либо же может быть использована встроенная в программный комплекс *nauty* программная утилита *findorb* [92].

Цвета графа будем представлять натуральными числами. На вход алгоритму подается неориентированный граф $G = (V, \alpha)$ и количество желаемых цветов $i \in \mathbb{N}$. В результате алгоритма будет получено множество S, которое представляет набор из попарно-неизоморфных векторов раскрасок графа.

Алгоритм 1.9.1. Генерация всех неизоморфных цветных графов методом МакКея.

Вход: $G = (V, \alpha)$, $i \in \mathbb{N}$.

1. Инициализируется множество S.

2. Вычисляются орбиты заданного графа orb_0.

3. Создается вектор раскраски col, в него добавляется цвет 1 для первой вершины. Здесь же инициализируются:

   a. $colsize = 1$ – размер col;

   b. $mc = 1$ – максимальный по значению цвет, находящийся в col;

   c. $colcnt = 0$ – количество найденных раскрасок.

4. Перебираются все цвета от 1 до $mc + 1$, очередной цвет присваивается cur.

5. Для cur перебираются элементы раскраски $col_j = \overline{1, colsize}$. Пусть $col[j] > cur$ и $orb_r[j] = orb_r[colsize + 1]$, где orb_r – орбиты, полученные разбиением вершин раскраской $r = (c_1, \dots, c_{j-1})$ – первые $(j - 1)$ цвета col. Тогда

данная раскраска объявляется изоморфной, отсекается и cur присваивается следующему цвету.

6. cur добавляется в col.

7. Вычисляется множество орбит orb_{col}, полученные разбиением вершин графа раскраской col.

8. Если $cur > mc$ и $mc \neq k$, то $mc = cur$.

9. Если $colsize < |V|$, то $colsize = colsize + 1$ и переход на шаг 3.

10. Иначе если $colsize = |V|$ и $mc = k$, то производится дополнительная проверка раскраски: для каждой вершины v_j проверяется выполнение условия:

$$\left(\forall x = \overline{1, \dots, (\iota - 1)}\right)\left(col_x > col_{v_j} \wedge orb_{col_{v_{x-1}}}(x) \neq orb_{col_{v_{x-1}}}(j)\right)$$

Если условие выполняется, то раскраска считается изоморфной и не включается в результирующее множество. Иначе необходимо добавить данную раскраску в результирующее множество и увеличить $colcnt$ на единицу.

11. Если не осталось неотсеченных раскрасок, то произвести выход из алгоритма.

В 10-м пункте производится дополнительная проверка раскраски с учетом изоморфизма цветов. Если же изоморфизм раскрасок рассматривается как изоморфизм с точностью до цветов, то пункты 7 и 10 алгоритма необходимо опустить. Вместо них после 9-го пункта необходимо выполнить следующую проверку:

Если $colsize = |V|$ и $mc = k$, то добавить текущую раскраску в результирующее множество.

Полученное на выходе множество S содержит все возможные попарно-неизоморфные вершинные раскраски заданного графа. Для того, чтобы получить попарно-неизоморфные цветные неориентированные графы, необходимо каждую раскраску из множества S сопоставить с графом G однозначно вычисляемой функцией раскраски f. Пусть есть вершинная раскраска $clr_j \in S$ и граф $G = (V, \alpha)$. Тогда функция для данной раскраски будет $f: v \to clr_j[v]$, где

функция f применима для любой вершины графа $\forall v \in V$, а $clr_j[v]$ обозначает v-ю позицию вектора clr_j.

Можно заметить, что данный алгоритм легко переложить для ориентированных графов, поскольку смежность вершин в нем не задействуется, и единственное отличие в алгоритмах заключается в разном построении множества орбит для неориентированного и ориентированного случаев графа.

В дальнейшем алгоритм 1.9.1 будем называть алгоритмом **Г-МК** (**Г**енерация методом **М**ак**К**ея).

1.10 Генерация цветных графов методом Рида-Фараджева

Теперь перейдем к рассмотрению переложения метода Рида-Фараджева на случай цветных графов.

В данном параграфе мы также будем рассматривать только *вершинные раскраски* неориентированного графа.

Как и в оригинальном методе Рида-Фараджева, здесь используется техника приближения для построения новых раскрасок графа. Целью выполнения генерации являются все возможные неизоморфные i-цветные раскраски заданного неориентированного графа. Предлагается построить алгоритм, позволяющий перебирать вершинные раскраски неориентированного графа без проверки на изоморфизм.

Идея алгоритма генерации заключается в следующем. На вход подается граф $G = (V, \alpha)$, ему сопоставляется функция раскраски $f: V \to F$. В начале алгоритма $F = \{1\}$, $|F| = 1$. То есть, поданному на вход графу сопоставляется вершинная раскраска $(1,1, \dots ,1,1)$. Для наглядного примера выберем цепь P_5,

представленную на рисунке 1.10.1. Белый цвет будет обозначать цвет 1 из заданного вектора вершинной раскраски.

Каждую итерацию алгоритма выбираем по одному представителю из множества орбит и последовательно сопоставляем каждому из данных представителей все возможные цвета от 1 до i.

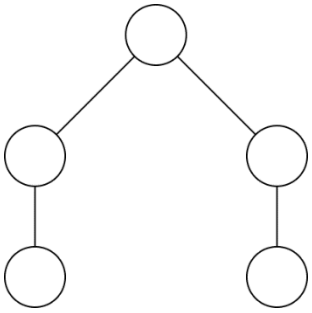

Рисунок 1.10.1 – Цепной граф P_5 с вершинной раскраской (1,1,1,1,1).

Допустим, в приведенном примере мы желаем построить все неизоморфные двухцветные графы. Тогда, определяя цвет 2 за красный, представим начальную итерацию алгоритма на рисунке 1.10.2.

Было обнаружено, что эффективнее выбирать наибольшего представителя из каждой орбиты. Сравнение представителей может быть произведено по нумерации вершин, либо каким-нибудь другим более подходящим способом.

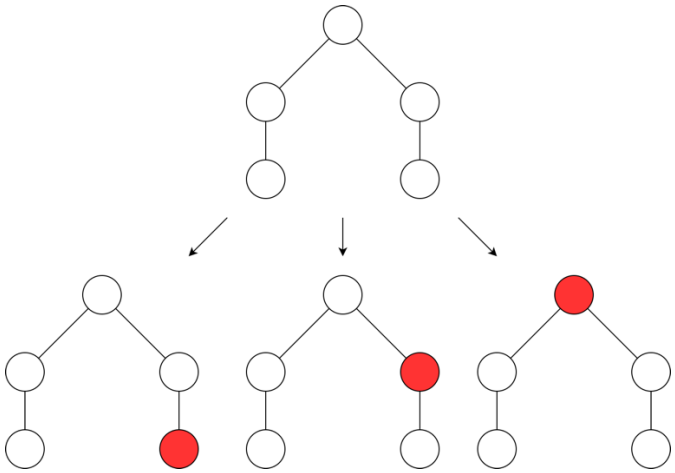

Рисунок 1.10.2 – Раскрашивание каждого представителя орбит P_5 во второй цвет.

Следующим шагом проверяется каноничность сгенерированных представителей. Элементы, соответствующие канонической нумерации, принима-

ются, все остальные – отсекаются. Выберем в качестве канонического родителя лексикографически наибольшего по вектору вершинной раскраски. Тогда, чтобы проверить представителя на каноничность, необходимо каждую вершину v из V последовательно раскрасить в цвета $F/\{f(v)\}$, и сравнить полученную раскраску с раскраской родителя данной раскраски. Если найдется какая-либо раскраска лексикографически меньше – значит данный граф уже встречался в дереве генерации, и, следовательно, его можно далее не рассматривать. Рассмотрим для примера уже введенный ранее граф P_5 с вершинной раскраской $(1,1,2,1,2)$. Тогда, следуя указанному принципу отсечения неканонических представителей, после перекраски графа получаются два возможных родителя: $(1,1,2,1,1)$ и $(1,1,1,1,2)$. Из этого следует, что если раскраска $(1,1,2,1,2)$ была порождена родителем $(1,1,1,1,2)$, то ее рассматривать не имеет смысла. Описанные действия изображены на рисунке 1.10.3 слева и справа соответственно. В первом случае раскраска $(1,1,2,1,2)$ принимается для дальнейшего рассмотрения, во втором – отсекается.

Каждый канонический представитель проверяется на количество уникальных цветов, и, если $|F| = i$ в данной раскраске, то она применяется в заданном графе G в виде некоторой функции раскраски f_j и сохраняется в результирующее множество неизоморфных цветных графов. Таким образом, на выходе получается множество попарно-неизоморфных цветных графов, построенных на основе техники канонических представителей методом Рида-Фараджева.

Подобный алгоритм позволяет существенно сократить количество перебираемых вариантов неизоморфных раскрасок. Можно заметить главное отличие данного метода от техники МакКея: если МакКей предлагает использовать итеративный подход к построению структур (в нашем случае раскрасок), то здесь производится поэтапное приближение на каждом уровне.

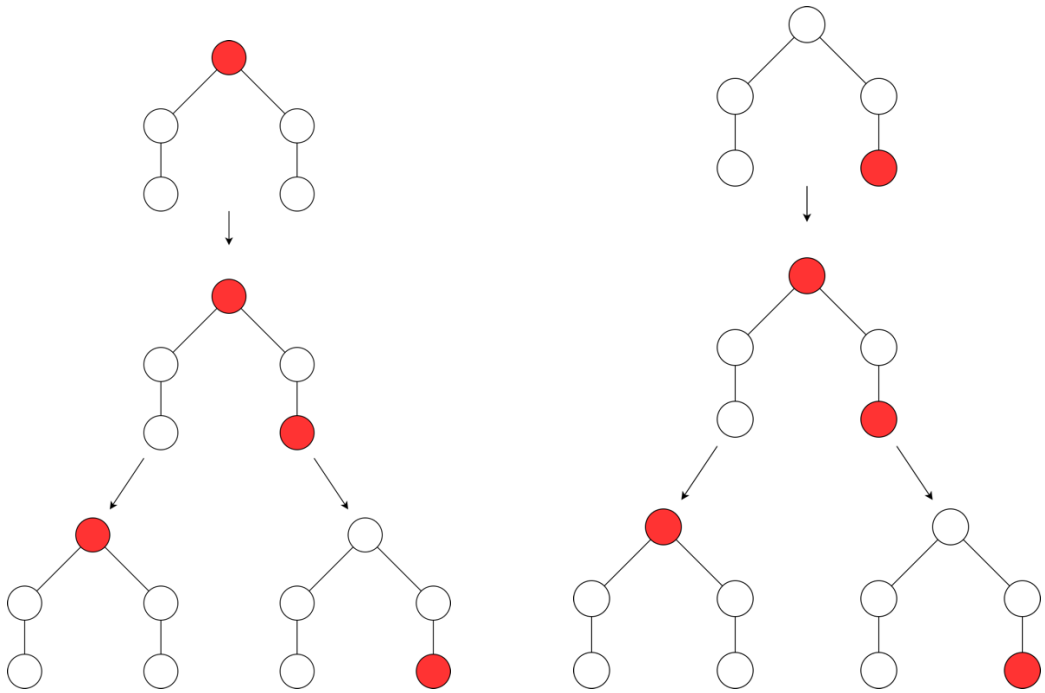

Рисунок 1.10.3 – Раскраска (1,1,2,1,2) слева является каноническим представителем, поскольку генерируется из лексикографически наименьшего родителя. Эта же раскраска справа каноническим представителем не является, поскольку существует родитель лексикографически меньше текущего.

Кроме того, на каждом уровне приближения производятся проверки, которые позволяют отсечь не интересующие в плане дальнейшего рассмотрения неканонические структуры.

1.11 Алгоритм генерации неизоморфных цветных графов методом Рида-Фараджева

Разработаем алгоритм на основании описанной в предыдущем параграфе технике генерации неизоморфных цветных графов без проверки на изоморфизм методом Рида-Фараджева.

Как и в предыдущем алгоритме, цвета графа будем представлять натуральными числами. Так же, как и в алгоритме Г-МК, на вход подается неориентированный граф $G = (V, \alpha)$ и количество желаемых цветов $i \in \mathbb{N}$. В результате алгоритма будет получено множество S – набор из попарно-неизоморфных неориентированных цветных графов.

Алгоритм 1.11.1. Генерация всех неизоморфных цветных графов методом Рида-Фараджева.

Вход: $G = (V, \alpha)$, $i \in \mathbb{N}$.

1. Инициализируется множество S.

2. Задается вектор $clr = (1,1, \dots ,1,1)$, где $|clr| = |V|$. Это инициализирующий вектор раскраски для поданного на вход графа G. clr_j обозначает j-й элемент вектора clr, то есть цвет j-й вершины графа.

3. По заданной разметке графа clr вычисляются его орбиты и записываются в orb.

4. Из каждой орбиты выбирается по одному наибольшему представителю o.

5. Цвет каждого представителя clr_o последовательно перекрашивается в цвета $\{clr_o + 1, \dots, i\}$.

6. Каждая clr, полученная перекрашиванием clr_o вершины o, проверяется на каноничность. Для проверки каноничности необходимо каждый элемент текущего clr последовательно раскрасить в цвета $\{1, \dots, clr_j\}$, и сравнить полученную раскраску с раскраской родителя данной раскраски. Если найдется какая-либо раскраска лексикографически меньше – значит данный граф уже встречался в дереве генерации, а значит, его можно далее не рассматривать. Очевидно, если очередной элемент clr_j уже имеет цвет 1, то смысла его перекрашивать и проверять нет, поэтому можно сразу переходить к следующему элементу clr_{j+1}.

7. Все принятые к дальнейшему рассмотрению раскраски добавляются в множество S.

8. Если не осталось ни одной неотсеченной раскраски, то алгоритм завершается.

9. По каждой неотсеченной раскраске строится новое множество орбит, и из каждой орбиты выбирается по одному наибольшему представителю o с той оговоркой, что выбранный представитель должен быть меньше представителя, выбранного на предыдущем шаге.

10. Переход на шаг 5.

Полученное на выходе множество S содержит все возможные попарно-неизоморфные вершинные раскраски заданного графа. Для того, чтобы получить попарно-неизоморфные цветные неориентированные графы, необходимо каждую раскраску из множества S сопоставить с графом G однозначно вычисляемой функцией раскраски f. Пусть есть вершинная раскраска $clr_j \in S$ и граф $G = (V, \alpha)$. Тогда функция для данной раскраски будет $f: v \to clr_j[v]$, где функция f применима для любой вершины графа $\forall v \in V$, а $clr_j[v]$ обозначает v-ю позицию вектора clr_j.

Как и алгоритм Г-МК, данный алгоритм без проблем может быть переложен на случай ориентированных графов. Здесь также никак не задействуется смежность вершин, а отличие будет в разности построения множества орбит.

Для удобства алгоритм 1.11.1 будем называть алгоритмом **Г-РФ** (**Г**енерация методом **Р**ида-**Ф**араджева).

1.12 Генерация неизоморфных графов с цветными ребрами

К сожалению, описанные выше алгоритмы Г-МК и Г-РФ довольно тяжело свести к случаю, когда необходимо построить множество попарно-неизоморфных графов с цветными ребрами. Тем не менее, получить способ генерировать графы с цветными ребрами нужно, поскольку графы с введенными на ней функциями реберной раскраски находят свое применение в различных сферах жизни.

Задача планирования процесса производства, или «*Open shop scheduling*» [74], задача расписания соединения для протокола связи множественного доступа с разделением по времени в беспроводных сенсорных сетях [71], задача назначения частоты света паре узлов в волоконно-оптической связи [68] – все эти проблемы могут быть переложены на графовую модель с цветными ребрами. Отдельной задачей следует выделить задачу разбиения расписания в круговых турнирах в несколько кругов, поскольку данный класс графов активно исследовался в смежных работах [6, 7] и других, то было бы полезно иметь некоторый фундамент для дальнейших исследований турниров с их переложением на случай графов с цветными ребрами.

Несмотря на то, что алгоритмы Г-МК и Г-РФ сложно применимы для графов с цветными ребрами, можно подойти к решению задачи генерации с другой стороны. Несложно заметить, что каждому графу G сопоставляется некоторый реберный граф $L(G)$, построенный по ребрам графа G.

Таким образом, предлагается следующий алгоритм для генерации попарно-неизоморфных графов с цветными ребрами с количеством цветов в точности i. Его идея состоит в том, чтобы сначала построить реберный граф, потом к нему применить алгоритм Г-МК или Г-РФ, а в конце перевести реберный граф обратно в исходный. Опять же, алгоритм рассматривает только неориентированные графы.

Алгоритм 1.12.1. Генерация всех неизоморфных графов с цветными ребрами.

Вход: $G = (V, \alpha)$, $i \in \mathbb{N}$.

1. Создается граф $L(G) = \left(V_{L(G)}, \alpha_{L(G)}\right)$, где $\left|V_{L(G)}\right| = 0$ и $\left|\alpha_{L(G)}\right| = 0$.

2. Для каждого ребра $(u, v) \in \alpha$ в $V_{L(G)}$ добавляется соответствующая этому ребру вершина с меткой (u, v), где $u < v$.

3. Для каждой вершины $(u, v) \in V_{L(G)}$ перебираются все остальные вершины из $(u', v') \in \mathrm{L}(G) \backslash \{(u, v)\}$: если $u = u'$ или $v = v'$, то в $\alpha_{L(G)}$ добавляется ребро $\left((u, v), (u', v')\right)$.

4. Таким образом, получен реберный граф $L(G)$, где каждая его вершина – некоторое ребро графа G, а ребро $L(G)$ есть некоторая вершина G. Теперь запустим алгоритм генерации неизоморфных цветных графов методом Мак-Кея (алгоритм Г-МК) или методом Рида-Фараджева (алгоритм Г-РФ), подавая на вход $L(G)$ и i.

5. Получив множество S вершинных раскрасок графа $L(G)$, для каждой раскраски найдем такую функцию, которая бы применяла рассматриваемый вектор раскраски на вершины $V_{L(G)}$. Данная функция будет универсальна для любого элемента множества S, и задается как функция $\psi: v_{L(G)} \to s_{v_{L(G)}}$, где $v_{L(G)} = (u', v')$ - некоторая вершина реберного графа с меткой (u', v'), а $s_{v_{L(G)}}$ – элемент вектора некоторой раскраски $s \in S$. Таким образом, если принимать метку (u', v') за обозначение ребра из α, то каждый элемент множества S может быть сопоставлен исходному графу G заданной функцией ψ: $G = (V, \alpha, \psi)$.

Для удобства алгоритм 1.12.1, использующий Г-МК будем обозначать **Г-МК-р**. А алгоритм 1.12.1, использующий Г-РФ – **Г-РФ-р**.

1.13 Вычислительные эксперименты для цветных цепей

Алгоритмы генерации неизоморфных цветных реализаций для заданного графа были реализованы для проведения различных вычислительных экспериментах и сбора данных. Реализация алгоритмов генерации включена в программный комплекс «*ColorGraphExtensions*» [37].

Были посчитаны все возможные раскраски для представляющих интерес классов графов с количеством вершин до 11. В работе [14] были рассмотрены сравнение времени работы метода МакКея и метода Рида-Фараджева для всех цепей и циклов с количеством вершин до 9. Также там приводится общее количество попарно-неизоморфных цветных графов, построенных заданной цепи или циклу.

Данная работа предлагает рассмотреть вычислительные результаты генерации цветных реализаций разными методами, а также уточнить общее время построения множества неизоморфных графов с цветами.

В таблице 1.13.1 представлен сравнительный анализ для цепных графов P_n для $n = \overline{2,11}$ и для $i = 2$. На рисунке 1.13.1 представлен пример цепного графа P_6. Случай $n = 1$ не имеет смысла рассматривать, поскольку P_1 содержит единственную раскраску при $|F| = 1$.

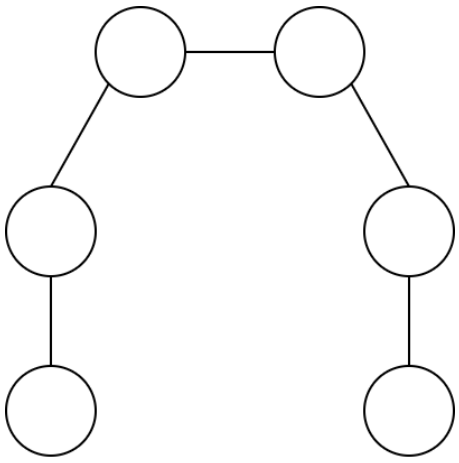

Рисунок 1.13.1 – цепной граф P_6.

Таблица 1.13.1 – Сравнительная таблица времени генерации цветных цепей $P_n = (V, \alpha, f)$ при $i = 2$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
2	1	0 мс.	0 мс.
3	3	2 мс.	2 мс.
4	6	3 мс.	2 мс.
5	13	5 мс.	5 мс.
6	27	10 мс.	7 мс.
7	55	11 мс.	8 мс.
8	115	13 мс.	12 мс.
9	231	15 мс.	12 мс.
10	479	35 мс.	21 мс.
11	959	79 мс.	39 мс.

В таблице 1.13.2 предлагается сравнение времени работы для графов P_n, $n = \overline{3, 11}$ при $i = 3$. В данной и каждой последующей таблице начальное количество вершин будет увеличиваться соразмерно числу цветов i. Таким образом, каждая таблица содержит $n = \overline{i, 11}$.

Таблица 1.13.2 – Сравнительная таблица времени генерации цветных цепей $P_n = (V, \alpha, f)$ при $i = 3$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
3	1	0 мс.	0 мс.
4	6	2 мс.	1 мс.
5	24	4 мс.	2 мс.
6	88	15 мс.	7 мс.
7	291	55 мс.	23 мс.
8	951	176 мс.	89 мс.
9	3199	630 мс.	274 мс.
10	9254	2 сек. 105 мс.	1 сек. 151 мс.
11	28241	7 сек. 249 мс.	3 сек. 439 мс.

Таблицы с 1.13.3 по 1.13.9 содержат результаты для графов P_n, $n = \overline{i, 11}$ при $i = \overline{4, 10}$.

Таблица 1.13.3 – Сравнительная таблица времени генерации цветных цепей $P_n = (V, \alpha, f)$ при $i = 4$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
4	1	0 мс.	0 мс.
5	10	25 мс.	9 мс.
6	65	104 мс.	50 мс.
7	348	490 мс.	201 мс.
8	1698	2 сек. 244 мс.	819 мс.
9	10942	11 сек. 114 мс.	3 сек. 832 мс.
10	34069	44 сек. 672 мс.	15 сек. 157 мс.
11	28241	3 мин. 23 сек. 662 мс.	1 мин. 19 сек. 775 мс.

Таблица 1.13.4 – Сравнительная таблица времени генерации цветных цепей $P_n = (V, \alpha, f)$ при $i = 5$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
5	1	0 мс.	0 мс.
6	15	478 мс.	161 мс.
7	140	2 сек. 793 мс.	905 мс.
8	1050	17 сек. 58 мс.	5 сек. 552 мс.
9	17890	1 мин. 34 сек. 786 мс.	28 сек. 7 мс.
10	42521	8 мин. 10 сек. 574 мс.	2 мин. 36 сек. 662 мс.
11	246674	25 мин. 18 сек. 402 мс.	13 мин. 3 сек. 581 мс.

Таблица 1.13.5 – Сравнительная таблица времени генерации цветных цепей $P_n = (V, \alpha, f)$ при $i = 6$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
6	1	0 мс.	0 мс.
7	120	11 сек. 777 мс.	3 сек. 141 мс.
8	2640	1 мин. 18 сек. 239 мс.	20 сек. 776 мс.
9	34560	8 мин. 46 сек. 580 мс.	2 мин. 15 сек. 901 мс.
10	352080	48 мин. 17 сек. 374 мс.	16 мин. 8 сек. 74 мс.
11	3091320	2 ч. 23 мин. 35 сек.	1 ч. 36 мин. 27 сек.

Таблица 1.13.6 – Сравнительная таблица времени генерации цветных цепей $P_n = (V, \alpha, f)$ при $i = 7$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
7	1	0 мс.	0 мс.
8	720	3 м. 24 сек. 604 мс.	1 мин. 10 сек. 335 мс.
9	20880	32 м. 48 сек. 98 мс.	9 мин. 7 сек. 341 мс.
10	353520	1 ч. 48 мин. 2 сек.	1 ч. 14 мин.
11	4587120	9 ч. 18 мин. 748 мс.	5 ч. 7 мин. 7 сек.

Таблица 1.13.7 – Сравнительная таблица времени генерации цветных цепей $P_n = (V, \alpha, f)$ при $i = 8$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
8	1	0 мс.	0 мс.
9	5040	1 ч. 28 м. 5 сек. 194 мс.	31 мин. 41 сек. 874 мс.
10	186480	4 ч. 12 мин. 36 сек.	2 ч. 40 мин. 1 сек.
11	3780000	7 ч. 42 мин. 32 сек.	4 ч. 18 мин. 7 сек.

Таблица 1.13.8 – Сравнительная таблица времени генерации цветных цепей $P_n = (V, \alpha, f)$ при $i = 9$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
9	1	0 мс.	0 мс.
10	40320	2 ч. 50 мин. 33 сек.	1 ч. 21 мин. 45 сек.
11	1530624	6 ч. 13 мин. 39 сек.	4 ч. 44 мин. 16 сек.

Таблица 1.13.9 – Сравнительная таблица времени генерации цветных цепей $P_n = (V, \alpha, f)$ при $i = 10$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
10	1	0 мс.	0 мс.
11	105960	10 ч. 40 мин. 49 сек.	6 ч. 34 мин. 58 сек.

Очевидно, что случай P_n при $i = 11$ рассматривать смысла не имеет, поскольку в результате получится единственный с точностью до изоморфизма 11-цветный цепной граф.

Стоит сделать замечание, что по сравнению с результатами, полученными ранее в работе [36], скорость работы реализаций обоих алгоритмов была улучшена путем оптимизации кода, что отражено в соответствующем программном комплексе [37].

Анализируя полученные данные, нужно отметить несколько важных свойств цветных графов и времени работы алгоритмов для них:

1. С увеличением количества цветов количество попарно-неизоморфных цветных графов P_n сначала увеличивается, достигая своего пика примерно на $n/2$, а далее уменьшается вплоть до 1 при $i = n$;

2. Алгоритм, основывающийся на итеративном подходе МакКея, работает медленнее для вычисления множества цветных цепей, чем аналогичный алгоритм, использующий метод приближения Рида-Фараджева.

1.14 Вычислительные эксперименты для цветных циклов

Следующим классом графов, представляющим интерес, являются циклы C_n. Для данного класса графов ранее были получены результаты работы алгоритмов Г-МК и Г-РФ для количества вершин до 9 [36]. Далее будут приведены результаты вычислительных экспериментов для количества вершин до 11.

Иллюстрация примера цикла представлена на рисунке 1.14.1.

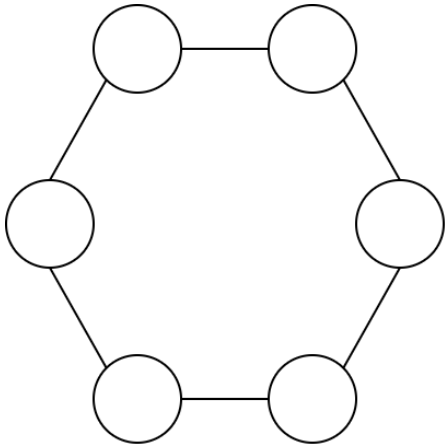

Рисунок 1.14.1 – Циклический граф C_6.

Таблицы 1.14.1–1.14.9 предлагают сравнительные данные времени работы алгоритмов на C_n, $n = \overline{i, 11}$, $i = \overline{2, 10}$. Как и в предыдущем параграфе, случай $i = 11$ рассматривать смысла не имеет, поскольку существует единственный с точностью до изоморфизма цветной цикл $C_{11} = (V, \alpha, f)$, у которого $|F| = 11$.

Таблица 1.14.1 – Сравнительная таблица времени генерации цветных циклов $C_n = (V, \alpha, f)$ при $i = 2$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
2	1	0 мс.	0 мс.
3	2	0 мс.	0 мс.
4	4	0 мс.	0 мс.
5	7	0 мс.	0 мс.

6	13	0 мс.	0 мс.
7	24	0 мс.	0 мс.
8	46	1 мс.	1 мс.
9	93	5 мс.	2 мс.
10	186	8 мс.	5 мс.
11	387	18 мс.	12 мс.

Таблица 1.14.2 – Сравнительная таблица времени генерации цветных циклов $C_n = (V, \alpha, f)$ при $i = 3$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
3	1	0 мс.	0 мс.
4	5	0 мс.	0 мс.
5	18	2 мс.	1 мс.
6	60	7 мс.	4 мс.
7	195	30 мс.	15 мс.
8	609	86 мс.	44 мс.
9	1934	350 мс.	167 мс.
10	5933	1 сек. 182 мс.	543 мс.
11	18438	4 сек. 88 мс.	2 сек. 251 мс.

Таблица 1.14.3 – Сравнительная таблица времени генерации цветных циклов $C_n = (V, \alpha, f)$ при $i = 4$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
4	1	0 мс.	0 мс.
5	9	19 мс.	7 мс.
6	54	69 мс.	30 мс.
7	281	409 мс.	145 мс.
8	1323	1 сек. 654 мс.	720 мс.
9	5997	7 сек. 688 мс.	2 сек. 855 мс.
10	25998	35 сек. 820 мс.	13 сек. 289 мс.
11	110909	2 мин. 35 сек. 592 мс.	55 сек. 15 мс.

Таблица 1.14.4 – Сравнительная таблица времени генерации цветных циклов $C_n = (V, \alpha, f)$ при $i = 5$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
5	1	0 мс.	0 мс.
6	14	375 мс.	141 мс.
7	125	2 сек. 443 мс.	1 сек. 12 мс.
8	907	13 сек. 23 мс.	4 сек. 273 мс.
9	5892	1 мин. 13 сек. 591 мс.	20 сек. 852 мс.
10	35488	6 мин. 16 сек. 548 мс.	1 мин. 51 сек. 409 мс.
11	203997	23 мин. 38 сек. 273 мс.	10 мин. 25 сек. 384 мс.

Таблица 1.14.5 – Сравнительная таблица времени генерации цветных циклов $C_n = (V, \alpha, f)$ при $i = 6$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
6	1	0 мс.	0 мс.
7	20	8 сек. 738 мс.	2 сек. 738 мс.
8	245	1 мин. 13 сек. 576 мс.	20 сек. 78 мс.
9	2380	8 мин. 48 сек. 34 мс.	1 мин. 57 сек. 588 мс.
10	20175	18 мин. 45 сек. 133 мс.	12 мин. 14 сек. 865 мс.
11	156644	2 ч. 33 мин. 58 сек.	1 ч. 20 мин. 32 сек.

Таблица 1.14.6 – Сравнительная таблица времени генерации цветных циклов $C_n = (V, \alpha, f)$ при $i = 7$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
7	1	0 мс.	0 мс.
8	720	2 мин. 1 сек. 394 мс.	1 мин. 5 сек. 236 мс.
9	20160	12 мин. 945 мс.	8 мин. 5 сек. 205 мс.
10	332640	1 ч. 32 мин. 12 сек.	58 мин. 14 сек. 575 мс.
11	4233600	10 ч. 34 мин. 58 сек.	7 ч. 8 мин. 2 сек.

Таблица 1.14.7 – Сравнительная таблица времени генерации цветных циклов $C_n = (V, \alpha, f)$ при $i = 8$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
8	1	0 мс.	0 мс.
9	5040	36 мин. 4 сек. 230 мс.	23 мин. 26 сек. 810 мс.
10	181440	4 ч. 12 мин. 56 сек.	3 ч. 35 мин. 12 сек.
11	3966480	6 ч. 14 мин. 57 сек.	4 ч. 55 мин. 37 сек.

Таблица 1.14.8 – Сравнительная таблица времени генерации цветных циклов $C_n = (V, \alpha, f)$ при $i = 9$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
9	1	0 мс.	0 мс.
10	40320	5 ч. 49 мин. 11 сек.	4 ч. 43 мин. 46 сек.
11	1429686	10 ч. 48 мин. 33 сек.	7 ч. 45 мин. 2 сек.

Таблица 1.14.9 – Сравнительная таблица времени генерации цветных циклов $C_n = (V, \alpha, f)$ при $i = 10$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
10	1	0 мс.	0 мс.
11	407208	12 ч. 34 мин. 11 сек.	9 ч. 58 мин. 29 сек.

Анализ полученных данных выделяет следующие свойства множества неизоморфных цветных циклических графов и времени работы алгоритмов для них:

1. С увеличением количества цветов количество попарно-неизоморфных цветных циклов C_n сначала увеличивается, достигая своего пика примерно на $n/2$, а далее уменьшается вплоть до 1 при $i = n$;

2. Алгоритм, основывающийся на итеративном подходе МакКея, работает медленнее для вычисления множества цветных циклов, чем аналогичный алгоритм, использующий метод приближения Рида-Фараджева.

3. При равнозначных условиях запуска программного комплекса вычисление цветных реализаций циклов алгоритм Рида-Фараджева работает примерно на 17% быстрее поиска цветных реализаций цепей при одинаковых n и i.

1.15 Вычислительные эксперименты для цветных полных графов

Еще одним классом графов, который будут рассмотрен в данной главе, является класс полных графов K_n. Данный класс примечателен тем, что общее количество ребер в полном n-вершинном графе равно $\frac{n(n-1)}{2}$, а значит, является довольно трудновычислимым. Тем не менее, в работе предоставлены сравнительные таблицы времени обработки заданного полного графа $K_n = (V, \alpha)$, $n = \overline{\imath, 11}$ при $i = \overline{2, 10}$.

Данный класс графов еще будет рассмотрен далее в работе. Пример полного шестивершинного графа представлен на рисунке 1.15.1.

Таблицы 1.15.1–1.15.9 содержат таблицы со сравнением времени работы алгоритмов Г-МК и Г-РФ для полных графов с количеством вершин до 11 и количеством цветов до 10.

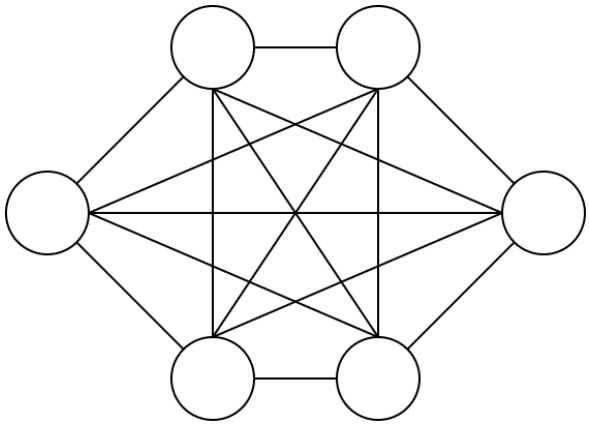

Рисунок 1.15.1 – Пример полного графа K_6.

Таблица 1.15.1 – Сравнительная таблица времени генерации полных цветных графов $K_n = (V, \alpha, f)$ при $i = 2$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
2	1	0 мс.	0 мс.
3	2	0 мс.	0 мс.
4	3	0 мс.	0 мс.
5	4	0 мс.	0 мс.
6	5	0 мс.	0 мс.
7	6	0 мс.	0 мс.
8	7	0 мс.	0 мс.
9	8	0 мс.	0 мс.
10	9	0 мс.	0 мс.
11	10	0 мс.	0 мс.

Таблица 1.15.2 – Сравнительная таблица времени генерации полных цветных графов $K_n = (V, \alpha, f)$ при $i = 3$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
3	1	0 мс.	0 мс.
4	2	0 мс.	0 мс.
5	8	0 мс.	0 мс.
6	22	2 мс.	2 мс.
7	52	5 мс.	4 мс.
8	114	11 мс.	9 мс.
9	240	25 мс.	22 мс.
10	494	55 мс.	50 мс.
11	1004	134 мс.	113 мс.

Таблица 1.15.3 – Сравнительная таблица времени генерации полных цветных графов $K_n = (V, \alpha, f)$ при $i = 4$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
4	1	0 мс.	0 мс.
5	6	7 мс.	4 мс.
6	42	22 мс.	16 мс.
7	192	68 мс.	57 мс.
8	732	200 мс.	179 мс.
9	2538	811 мс.	622 мс.
10	8334	2 сек. 55 мс.	1 сек. 921 мс.
11	26484	7 сек. 112 мс.	6 сек. 486 мс.

Таблица 1.15.4 – Сравнительная таблица времени генерации полных цветных графов $K_n = (V, \alpha, f)$ при $i = 5$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
5	1	0 мс.	0 мс.
6	24	145 мс.	75 мс.
7	264	484 мс.	346 мс.
8	1824	2 сек. 139 мс.	1 сек. 557 мс.
9	10224	11 сек. 392 мс.	7 сек. 248 мс.
10	51048	45 сек. 283 мс.	31 сек. 366 мс.
11	237528	2 мин. 59 сек. 742 мс.	2 мин. 18 сек. 20 мс.

Таблица 1.15.5 – Сравнительная таблица времени генерации полных цветных графов $K_n = (V, \alpha, f)$ при $i = 6$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
6	1	0 мс.	0 мс.
7	120	3 сек. 12 мс.	1 сек. 679 мс.
8	1920	12 сек. 347 мс.	8 сек. 894 мс.
9	18720	1 мин. 2 сек. 674 мс.	47 сек. 833 мс.
10	144720	6 мин. 11 сек. 9 мс.	4 мин. 26 сек. 52 мс.
11	978840	34 мин. 3 сек. 501 мс.	16 мин. 54 сек. 621 мс.

Таблица 1.15.6 – Сравнительная таблица времени генерации полных цветных графов $K_n = (V, \alpha, f)$ при $i = 7$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
7	1	0 мс.	0 мс.
8	720	54 сек. 282 мс.	39 сек. 969 мс.
9	15840	6 мин. 34 сек. 293 мс.	4 мин. 7 сек. 15 мс.
10	207360	58 мин. 28 сек. 596 мс.	35 мин. 49 сек. 239 мс.
11	775520	1 ч. 13 мин. 22 сек.	48 мин. 11 сек. 5 мс.

Таблица 1.15.7 – Сравнительная таблица времени генерации полных цветных графов $K_n = (V, \alpha, f)$ при $i = 8$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
8	1	0 мс.	0 мс.
9	5040	48 мин. 561 мс.	18 мин. 23 сек. 968 мс.
10	97956	8 ч. 5 мин. 58 сек.	5 ч. 42 мин. 19 сек.
11	223152	13 ч. 48 мин.	10 ч. 29 мин. 56 сек.

Таблица 1.15.8 – Сравнительная таблица времени генерации полных цветных графов $K_n = (V, \alpha, f)$ при $i = 9$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
9	1	0 мс.	0 мс.
10	7944	8 ч. 5 мин. 7 сек.	6 ч. 45 мин. 30 сек.
11	18960	12 ч. 49 мин. 5 сек.	10 ч. 48 мин. 58 сек.

Таблица 1.15.9 – Сравнительная таблица времени генерации полных цветных графов $K_n = (V, \alpha, f)$ при $i = 10$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
10	1	0 мс.	0 мс.
11	1549	13 ч. 3 мин. 59 сек.	11 ч. 45 мин. 22 сек.

Анализ приведенных таблиц позволяет выделить следующие замечания для цветных полных графов:

1. С увеличением количества цветов количество попарно-неизоморфных цветных полных графов K_n сначала увеличивается, достигая своего пика на $i = 5$, а далее уменьшается вплоть до 1 при $i = n$;

2. Алгоритм, основывающийся на итеративном подходе МакКея, работает медленнее для вычисления множества цветных полных графов, чем аналогичный алгоритм, использующий метод приближения Рида-Фараджева.

1.16 Вычислительные эксперименты для цветных звезд

Последним рассматриваемым классом графов, для которых построение неизоморфных цветных графов представляет интерес, является класс звездных графов. Он представляет особый интерес в поиске расширений, что будет описано далее в подробностях.

Пример шестивершинной звезды представлены на рисунке 1.16.1.

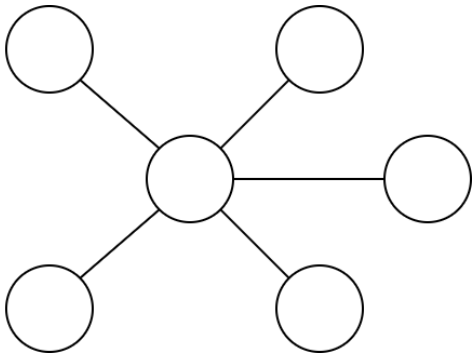

Рисунок 1.16.1 – Пример звездного графа S_6.

Реализации алгоритмов генерации были запущены на звездных графах S_n с количеством вершин до 11, при этом количество цветов в графе было рас-

смотрено до 10. Таким образом, таблицы 1.16.1–1.16.9 предоставляют результаты сравнения времени работы алгоритмов на графах $S_n = (V, \alpha)$, где $n = \overline{i, 11}$, $i = \overline{2, 10}$.

Таблица 1.16.1 – Сравнительная таблица времени генерации звездных цветных графов $S_n = (V, \alpha, f)$ при $i = 2$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
2	1	0 мс.	0 мс.
3	2	0 мс.	0 мс.
4	3	0 мс.	0 мс.
5	4	0 мс.	0 мс.
6	5	0 мс.	0 мс.
7	6	0 мс.	0 мс.
8	7	0 мс.	0 мс.
9	8	0 мс.	0 мс.
10	9	0 мс.	0 мс.
11	10	0 мс.	0 мс.

Таблица 1.16.2 – Сравнительная таблица времени генерации звездных цветных графов $S_n = (V, \alpha, f)$ при $i = 3$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
3	1	0 мс.	0 мс.
4	2	0 мс.	0 мс.
5	8	1 мс.	1 мс.
6	22	3 мс.	2 мс.
7	52	8 мс.	6 мс.
8	114	18 мс.	13 мс.
9	240	32 мс.	30 мс.
10	494	91 мс.	67 мс.
11	1004	214 мс.	198 мс.

Таблица 1.16.3 – Сравнительная таблица времени генерации звездных цветных графов $S_n = (V, \alpha, f)$ при $i = 4$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
4	1	0 мс.	0 мс.
5	6	10 мс.	6 мс.
6	42	24 мс.	20 мс.
7	192	101 мс.	67 мс.
8	732	400 мс.	234 мс.
9	2538	1 сек. 12 мс.	751 мс.
10	8334	3 сек. 335 мс.	2 сек. 615 мс.
11	26484	10 сек. 55 мс.	8 сек. 189 мс.

Таблица 1.16.4 – Сравнительная таблица времени генерации звездных цветных графов $S_n = (V, \alpha, f)$ при $i = 5$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
5	1	0 мс.	0 мс.
6	24	484 мс.	111 мс.
7	264	693 мс.	432 мс.
8	1824	3 сек. 17 мс.	1 сек. 766 мс.
9	10224	10 сек. 348 мс.	7 сек. 421 мс.
10	51048	48 сек. 458 мс.	38 сек. 763 мс.
11	237528	3 мин. 9 сек. 112 мс.	2 мин. 44 сек. 620 мс.

Таблица 1.16.5 – Сравнительная таблица времени генерации звездных цветных графов $S_n = (V, \alpha, f)$ при $i = 6$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
6	1	0 мс.	0 мс.
7	120	3 сек. 349 мс.	1 сек. 743 мс.
8	1920	15 сек. 104 мс.	10 сек. 203 мс.
9	18720	1 мин. 6 сек. 684 мс.	55 сек. 235 мс.
10	144720	8 мин. 12 сек. 922 мс.	5 мин. 6 сек. 284 мс.
11	978840	59 мин. 45 сек.	36 мин. 394 мс.

Таблица 1.16.6 – Сравнительная таблица времени генерации звездных цветных графов $S_n = (V, \alpha, f)$ при $i = 7$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
7	1	0 мс.	0 мс.
8	720	3 мин. 34 сек. 44 мс.	1 мин. 22 сек. 781 мс.
9	15840	5 мин. 46 сек. 676 мс.	3 мин. 28 сек. 455 мс.
10	207360	18 мин. 10 сек. 394 мс.	12 мин. 45 сек. 182 мс.
11	775520	1 ч. 58 мин. 49 сек.	1 ч. 22 мин. 45 сек.

Таблица 1.16.7 – Сравнительная таблица времени генерации звездных цветных графов $S_n = (V, \alpha, f)$ при $i = 8$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
8	1	0 мс.	0 мс.
9	5040	44 мин. 11 сек. 132 мс.	28 мин. 2 сек. 464 мс.
10	83808	1 ч. 58 мин. 38 сек.	1 ч. 24 мин. 104 мс.
11	188630	2 ч. 59 мин. 37 сек.	2 ч. 28 мин. 50 сек.

Таблица 1.16.8 – Сравнительная таблица времени генерации звездных цветных графов $S_n = (V, \alpha, f)$ при $i = 9$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
9	1	0 мс.	0 мс.
10	7944	1 ч. 33 мин. 2 сек.	43 мин. 47 сек. 866 мс.
11	18960	4 ч. 59 мин. 6 сек.	3 ч. 45 мин. 38 сек.

Таблица 1.16.9 – Сравнительная таблица времени генерации звездных цветных графов $S_n = (V, \alpha, f)$ при $i = 10$ различными методами

Количество вершин	Количество раскрасок	Время работы Г-МК	Время работы Г-РФ
10	1	0 мс.	0 мс.
11	1549	7 ч. 4 мин. 19 сек.	5 ч. 18 мин. 33 сек.

Анализ полученных данных позволил выявить совпадение результатов для цветных полных графов и цветных звезд. Общее время генерации цветных реализаций для звезд на порядок меньше, чем генерация цветных реализаций для полных графов при аналогичных входных данных.

1.17 Вычислительные эксперименты для графов с цветными ребрами

Необходимо заметить, что для следующих рассмотренных классов графов можно найти закономерности между множествами неориентированных цветных графов и неориентированных графов с цветными ребрами. Так, после проведения вычислительных экспериментов с запуском реализаций алгоритмов Г-МК-р и Г-РФ-р было обнаружено:

1. Для цепного графа P_n алгоритмы генерации цветных реберных реализаций строят реберный граф $L(P_n)$, в точности являющимся графом P_{n-1}. Наглядный пример расположен на рисунке 1.17.1.

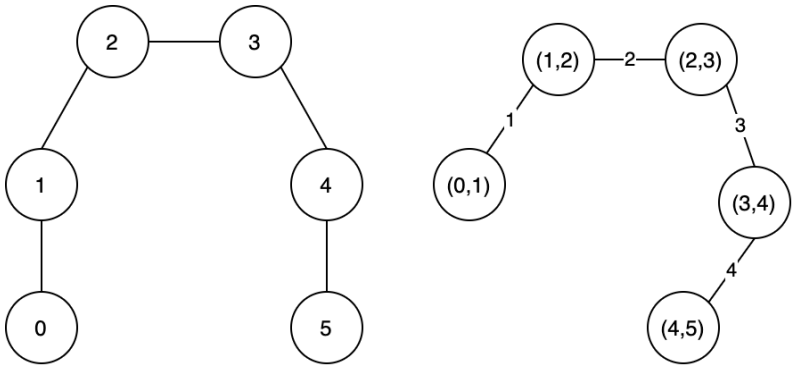

Рисунок 1.17.1 – Цепь P_6 и ее реберный граф $L(P_6)$.

2. Для графа-цикла C_n алгоритмы генерации цветных реберных реализаций строят реберный граф $L(C_n)$, в точности являющимся графом C_n. Пример реберного графа для цикла C_6 представлен на рисунке 1.17.2.

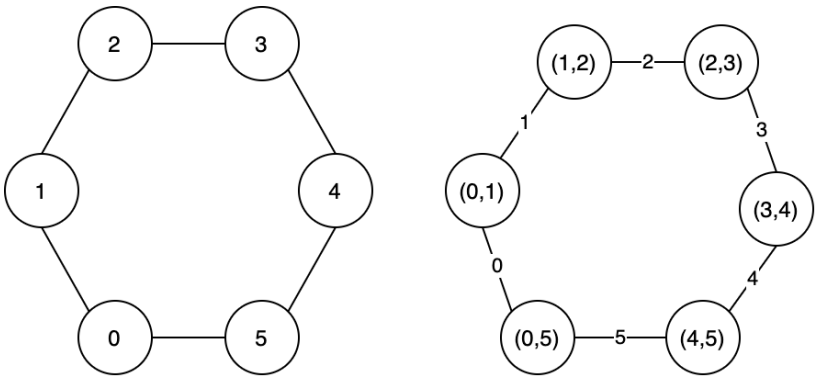

Рисунок 1.17.2 – Цикл C_6 и его реберный граф $L(C_6)$.

3. Для полного графа $K_n = (V, \alpha)$ алгоритмы генерации цветных реберных реализаций строят реберный граф $L(K_n)$, в точности являющимся регулярным графом $R_{|\alpha|,2(n-2)}$. Пример реберного графа, построенного по графу K_4 можно найти на рисунке 1.17.3.

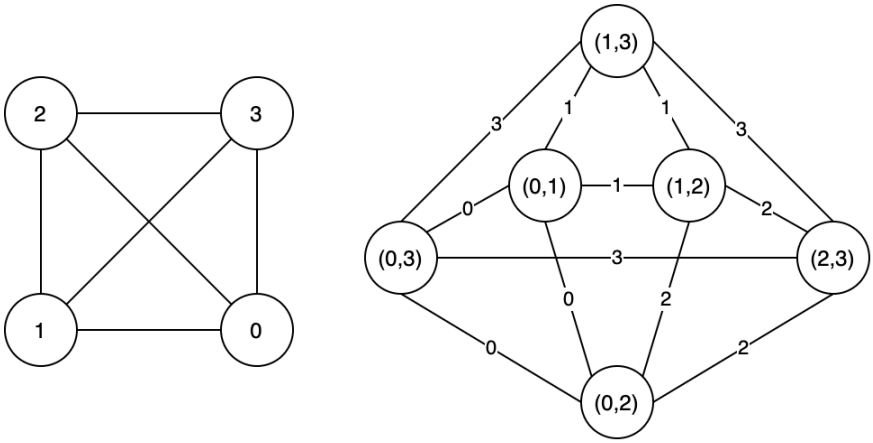

Рисунок 1.17.3 – Пример полного графа K_6 и его реберного расширения $L(K_6)$.

4. Для звездного графа $S_n = (V, \alpha)$ алгоритмы генерации цветных реберных реализаций строят реберный граф $L(S_n)$, в точности являющимся полным графом K_{n-1}. Пример изображен на рисунке 1.17.4.

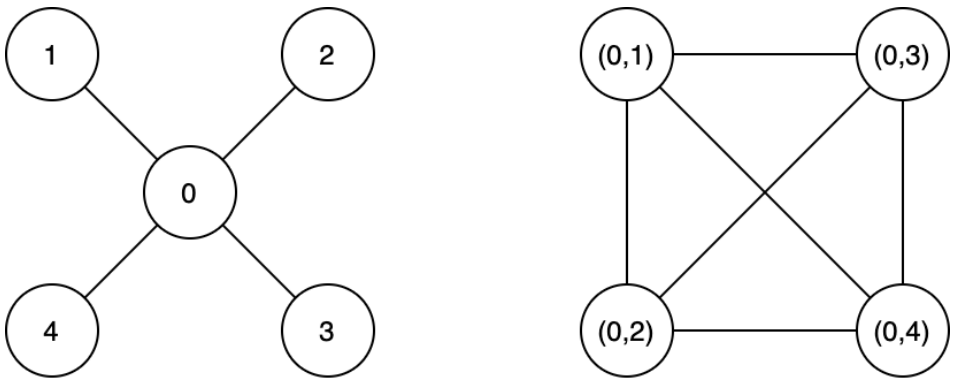

Рисунок 1.17.4 – Пример звезды S_5 и ее реберного графа $L(S_5)$.

Из этого следует, что:

1. Мощность множества неизоморфных цветных цепей P_n эквивалентна мощности множества неизоморфных цепей P_{n+1} с цветными ребрами. Множество вершинных раскрасок для реберного графа $L(P_n)$ и множество вершинных раскрасок для графа P_{n-1} совпадают.

2. Мощность множества неизоморфных цветных циклов C_n эквивалентна мощности множества неизоморфных циклов C_n с цветными ребрами. Множество вершинных раскрасок для реберного графа $L(C_n)$ и множество вершинных раскрасок для графа C_n совпадают.

3. Мощность множества неизоморфных цветных полных графов K_n эквивалентна мощности множества неизоморфных звездных графов S_{n+1} с цветными ребрами. Множество вершинных раскрасок для реберного графа $L(S_n)$ и множество вершинных раскрасок для графа K_{n-1} совпадают.

4. Мощность множества неизоморфных цветных регулярных графов $R_{\frac{n(n-1)}{2},2(n-2)}$ эквивалентна мощности множества неизоморфных полных графов K_n с цветными ребрами. Множество вершинных раскрасок для реберного графа $L(K_n)$ и множество вершинных раскрасок для графа $R_{\frac{n(n-1)}{2},2(n-2)}$ совпадают.

Все озвученные варианты (кроме одного) тем или иным образом уже рассматривались во время предыдущих вычислительных экспериментов. Единственным случаем является генерация неизоморфных полных графов с цветными ребрами. Поскольку из предыдущих запусков становится понятно, что при росте количества вершин время работы реализации экспоненциально растет, и генерация для графа с количеством вершин больше 10 является трудоемкой задачей, то приведем сравнительную таблицу для полных графов с цветными ребрами на относительно небольших полных графах.

Таблица 1.17.1 – Сравнительная таблица времени генерации полных графов с цветными ребрами различными методами

Количество цветов	Количество вершин	Количество раскрасок	Время работы Г-МК-р	Время работы Г-РФ-р
2	3	2	0 мс.	0 мс.
	4	7	0 мс.	0 мс.
	5	179	7 мс.	4 мс.
	6	16383	1 сек. 184 мс.	692 мс.
3	4	43	50 мс.	27 мс.
	5	7547	2 сек. 900 мс.	721 мс.
	6	2375101	15 мин. 3 сек. 456 мс.	4 мин. 55 сек. 249 мс.
4	5	33710	44 сек. 948 мс.	15 сек. 333 мс.
	6	82353168	5 ч. 22 мин. 56 сек.	3 ч. 12 мин. 36 сек.
5	6	433783368	16 ч. 3 мин. 59 сек.	11 ч. 34 мин. 8 сек.

Полученные результаты были собраны в базу данных множества полных графов с цветными ребрами.

Глава 2. Минимальные расширения цветных графов

Данная глава содержит решение задачи поиска минимальных расширений для заданных цветных графов. В главе описывается разработка методов и алгоритмов поиска минимальных расширений для цветных графов. Разработанные алгоритмы реализованы и включены в программный комплекс *ColorGraphExtensions* [37]. Проведены вычислительные эксперименты с использованием данной реализации. Данные, полученные в ходе вычислительных экспериментов, далее в работе будут использованы для проверки аналитических результатов для заданных классов графов.

Определения, использованные в этой главе, соответствуют работам [1] и [39, 46, 78, 79].

2.1 Отказоустойчивость в теории графов

Одной из наиболее актуальных задач настоящего времени является проблема проектирования технической системы, которая бы позволяла сохранять целостность данных, непрерывность работы и доступа к этим данным. Одним из наиболее простых решений, и поэтому распространенным, является обеспечение технической системы свойством *высокой доступности* (highly availability, HA) [102]. Данное свойство позволяет гарантировать пользователям доступ без незапланированных простоев, а временные рамки простоя (время обслуживания системы) заранее объявлять.

Однако, что, если требуется внедрить систему в такую сферу жизни, где даже краткосрочный простой может оказать негативное влияние всю работу или результат? Например, при простое в сфере электроснабжения, коммерции или даже космической отрасли самым невинным негативным результатом будет являться потеря дохода, и масштаб данной потери зависит от времени, которое техническая система провела в бездействии. В подобных ситуациях применяются более сложные системы с повышенным свойством надежности. Одним из первых ученых, кто обратил внимание на задачу повышения надежности вычислительных систем, является А. Авиженис. В своей работе [46] он предложил два различных подхода к повышению надежности:

1. предотвращение ошибок путем внедрения высоконадежных компонентов системы;

2. обеспечение свойства отказоустойчивости, смысл которого в том, чтобы ввести в систему некоторые избыточные компоненты, которые в случае отказа могли бы заменить отказавшие в работе части системы.

Само понятие отказоустойчивости и смежных с ним терминов Авиженис трактовал следующим образом.

Определение 2.1.1. Отказоустойчивость (fault tolerance) есть обеспечение системы способностью противостоять ошибке и возможностью продолжать работу в присутствии этой ошибки.

Определение 2.1.2. Отказ означает, что предоставляемые системой услуги отличаются от правильных. Другими словами, *отказ* означает, что одно или несколько состояний системы отличаются от ожидаемого поведения при правильном обслуживании.

Определение 2.1.3. Отличие ожидаемого поведения системы от наблюдаемого называется *ошибкой.*

Определение 2.1.4. Причина ошибки, будь она обнаруженной или гипотетической, называется *сбоем* (*fault*).

Следующим витком исследований отказоустойчивости стали работы [40, 41, 47, 84, 85] на тему проектирования гарантоспособных систем. В подобных работах предлагается усовершенствовать свойство надежности системы до свойства гарантоспособности. Под гарантоспособностью в этом смысле понимают следующее.

Определение 2.1.5. Гарантоспособностью называют комплексное свойство системы предоставлять требуемые услуги, которым можно оправданно доверять. Гарантоспособность включает в себя следующие свойства:

– *готовность* – готовность к правильному обслуживанию;

– *надежность* – непрерывность правильного обслуживания;

– *безопасность* – отсутствие катастрофических последствий для пользователей и окружающей среды;

– *целостность* – отсутствие некорректных изменений системы;

– *обслуживаемость* – способность подвергаться модификациям и ремонту.

Угрозами гарантоспособности являются ошибки, отказы и сбои системы.

Именно отказоустойчивость позволяет поддерживать непрерывное правильное обслуживание и целостность данных. Выделяют два уровня полной отказоустойчивости:

1. *амортизация отказов* – система продолжает работать в присутствии ошибок с частичной деградацией функциональных возможностей;

2. *полная отказоустойчивость* – система продолжает работать в присутствии ошибок без существенной потери функциональных свойств.

Следующим этапом развития исследований отказоустойчивости и гарантоспособных систем стало переложение описанной выше проблемы на графовую модель. Джон Хейз [79] предложить рассматривать задачу полной отказоустойчивости технических систем на сопоставленных им графах.

Рассмотрим графовую модель отказоустойчивой технической системы. Некоторой технической системе Σ сопоставляется помеченный граф $G(\Sigma) =$

(V, α). Множеством V в данном графе являются элементы системы Σ, а ребрами из множества α – связи между этими элементами. Метки вершин обозначают тип элементов. Тогда под *полным отказом* элемента системы Σ понимают удаление вершины из V, соответствующей отказавшему элементу системы, и всех инцидентных ей ребер.

Позднее Д. Хейз совместно с известным американским математиком Фрэнком Харари [77] обобщили модель на случай отказа связей между элементами, назвав свойство системы продолжать работу в присутствии подобных сбоев *реберной отказоустойчивостью.* Из этого следовало, что отказы элементов можно называть *вершинной отказоустойчивостью* [78].

Таким образом, в данных работах были предложены термины *вершинной* и *реберной k-отказоустойчивой реализации* технической системы.

Определение 2.1.6. Говорят, что система Σ^* называется *вершинной k-отказоустойчивой реализацией* системы Σ, или просто *k-отказоустойчивой реализацией*, если отказ любых k элементов системы Σ^* приводит к графу, в который можно вложить граф $G(\Sigma)$, сопоставленный системе Σ.

При этом Хейз предлагал такую конфигурацию системы Σ^*, при которой в нормальном режиме работы (то есть без присутствия ошибок), избыточные элементы маскируются, а при сбое происходит реконфигурация, которая позволит системе Σ продолжать работу без потери функциональных возможностей.

Аналогичное определение вводится и для *реберной k-отказоустойчивой реализации* системы Σ.

Определение 2.1.6 подходит для систем с однотипными элементами. На практике подобные системы встречаются довольно часто, например, это могут быть массивно параллельные вычисления. Здесь элементом системы является комбинация одного или нескольких процессоров, локальной памяти и других компонент. В случае же технической системы с элементами разного типа не все так однозначно. При построении однотипной вершинной k-отказоустойчивой реализации не учитываются особенности разнотиповых систем, из-за

чего отказ какого-либо элемента некоторого типа может быть фатален, если он не продублирован избыточным элементом того же типа.

Кроме того, отказоустойчивая реализация может содержать лишнее количество избыточных связей между элементами, что на практике приведет к увеличению стоимости построения подобной системы. Поэтому предлагается определение с более строгими условиями для отказоустойчивой реализации.

Определение 2.1.7. k-отказоустойчивая реализация Σ^* системы Σ, содержащей элементы t различных типов, называется *оптимальной*, если система Σ^* отличается от системы Σ на k элементов каждого из t типов системы Σ и среди всех подобных отказоустойчивых реализаций с тем же числом элементов система Σ^* обладает наименьшим числом связей.

В работе [79] Хейз рассмотрел отказоустойчивые реализации для цепей, циклов и помеченных деревьев.

Позднее М. Б. Абросимов в своей работе [1] предложил более лаконичное название для отказоустойчивых реализаций систем Σ, переложенных на графовую модель – *вершинные* и, соответственно, *реберные расширения* графов. В настоящей работе все дальнейшие исследования и результаты используют именно эту терминологию, соответственно, необходимо сформулировать все требуемые понятия в контексте расширений графов.

2.2 Расширения графов

Дадим основные определения главы 2 в соответствии с [1].

Определение 2.2.1. Граф $G^* = (V^*, \alpha^*)$ называется *вершинным k-расширением* графа $G = (V, \alpha)$, если граф G вкладывается в каждый граф, получающийся удалением любых k вершин из графа G^* со всеми инцидентными ему ребрами.

Определение 2.2.2. Граф $G^* = (V^*, \alpha^*)$ называется *реберным k-расширением* графа $G = (V, \alpha)$, если граф G вкладывается в каждый граф, получающийся удалением любых k ребер из графа G^*.

Вершины, содержащиеся в графе G^*, но не в G, будем называть *дополнительными вершинами*, а ребра – *дополнительными ребрами.*

Определение 2.2.3. Тривиальным называется расширение G^* графа G такое, что дополнительные вершины графа G^* соединены ребрами между собой и со всеми вершинами графа G.

Определение 2.2.4. Граф $G^* = (V^*, \alpha^*)$ графа $G = (V, \alpha)$ называют *минимальным вершинным k-расширением*, если он удовлетворяет следующим условиям:

1. граф G^* является вершинным k-расширением графа G;
2. граф G^* содержит в точности на k вершин больше, чем граф G: $|V^*| = |V| + k$;
3. среди всех графов, удовлетворяющих первым двум условиям, α^* содержит наименьшее количество ребер.

Аналогично вводится понятие минимального реберного расширения.

Определение 2.2.5. Граф $G^* = (V^*, \alpha^*)$ графа $G = (V, \alpha)$ называется *минимальным реберным k-расширением*, если он удовлетворяет следующим условиям:

1. граф G^* является реберным k-расширением графа G;
2. граф G^* содержит в точности столько же вершин, что и граф G: $|V^*| = |V|$;
3. среди всех графов, удовлетворяющих первым двум условиям, граф G^* содержит наименьшее количество ребер в α^*.

Несложно заметить, что данные условия будут выполняться только для непомеченных графов, поскольку иначе нарушается условие вложенности графа с различными метками в получающийся граф при отказе элементов или их связей. В данной работе рассматриваются помеченные графы, то есть цветные графы. Поэтому кажется резонным перенести терминологию расширений на случай цветных графов.

2.3 Терминология расширений цветных графов

Для цветных графов следует переопределить операцию вложения.

Определение 2.3.1. Говорят, что цветной граф $G_1 = (V_1, \alpha_1, f_1)$ *допускает вложение* цветного графа $G_2 = (V_2, \alpha_2, f_2)$, если:

1. существует взаимно однозначное отношение $\zeta: V_2 \to V_1$, которое позволяет вложить G_2 в граф G_1: $\forall\, u, v \in V: (u, v) \in \alpha_1 \Leftrightarrow \big(\zeta(u), \zeta(v)\big) \in \alpha_2$;
2. цвет любой вершины $v \in G_2$ сохраняется при вложении:

$$\forall\, v \in G_2: f_1(v) = f_2\big(\zeta(v)\big).$$

Здесь отношение вложения, введенное в первом параграфе первой главы, было обозначено через ζ для удобства восприятия.

Также часто говорят, что граф G_1 допускает вложение графа G_2 *с учетом цветов.*

Введенная операция позволяет переформулировать понятие *оптимальной отказоустойчивой системы* в контексте теории графов. Итак, переложим следующие понятия на случай цветных графов.

Определение 2.3.2. Граф $G^* = (V^*, \alpha^*, f^*)$ называется *вершинным k-расширением* цветного графа $G = (V, \alpha, f)$, если граф G можно вложить *с учетом*

цветов в каждый граф, полученный удалением любых k вершин графа G^* со всеми инцидентными им ребрами.

Аналогично для реберного расширения необходимо учитывать сохранность цветов при его построении.

Определение 2.3.3. Граф $G^* = (V^*, \alpha^*, f^*)$ называется *минимальным вершинным k-расширением* i-цветного графа $G = (V, \alpha, f)$, если выполняются следующие условия:

1. граф G^* является вершинным k-расширением цветного графа G;
2. V^* содержит на ik больше вершин, чем V: $|V^*| = |V| + ik$;
3. среди всех графов, удовлетворяющих условиям 1 и 2, α^* имеет наименьшее количество ребер.

Очевидно, что минимальное количество дополнительных вершин должно быть равно произведению количества цветов i на мощность отказоустойчивости k графа G^*. Допустим, в графе G множество количества вершин различных цветов обозначается как $V_f = \{|V_0|, |V_1|, \dots, |V_j|, \dots, |V_i|\}$, где $V_j = \{f(v) = j : \forall\, v \in V\}$ – все вершины цвета j. Тогда, если в G^* количество дополнительных вершин будет меньше ik хотя бы на единицу, то множество $V_f = \{|V_0| + 1, |V_1| + 1, \dots, |V_{j-1}| + 1, |V_j|, |V_{j+1}| + 1, \dots, |V_i| + 1\}$, а значит при удалении любой вершины цвета j количество вершин этого цвета становится равным $|V_j| - 1$, то есть условие сохранности цветов нарушается, и G^* не может являться вершинным расширением цветного графа G.

Определение *минимального реберного k-расширения* аналогично данному в предыдущем параграфе с той оговоркой, что первый пункт определения, то есть «G^* является реберным k-расширением цветного графа G», необходимо рассматривать *с учетом сохранности цветов*.

Надо отметить, что определения минимальных расширений цветных графов являются переложением термина оптимальной отказоустойчивой реализации на модель графов.

2.4 Задача поиска минимальных расширений цветных графов

Задача поиска минимальных расширений для заданного графа представляет особый интерес при практическом применении. Развивая мысль поиска и построения отказоустойчивых реализаций технических систем, минимальные расширения графов позволяют найти ответы на следующие вопросы:

– Возможно ли спроектировать техническую систему с непрерывным доступом ко всем предоставляемым функциональным возможностям?

– Как минимизировать затраты на создание подобной системы?

– Как спроектировать подобную техническую систему?

Ответы на данные вопросы, как уже было сказано, Д. Хейз предложил искать на модели графов. Он же предложил первые схемы построения минимальных вершинных 1-расширений для цепей и циклов, а также для частного случая помеченного дерева [79].

Довольно быстро было доказано, что задача поиска минимальных расширений является вычислительно сложной. Сложность поиска заключается в том, что проверка, является ли некоторый граф вершинным k-расширением заданного графа, относится к NP-полным задачам [3]. В работах [1, 2] предлагается следующий переборный алгоритм поиска всех минимальных вершинных k-расширений.

Алгоритм 2.4.1. Построение всех минимальных вершинных k-расширений заданного графа G, отличного от вполне несвязного.

Вход: $G = (V, \alpha)$, $k \in \mathbb{N}$.

1. $m := 0$.

2. $m := m + 1$.

3. Строятся все графы, полученные добавлением k вершин и m ребер в граф G.

4. Среди построенных на предыдущем шаге графов выбираются вершинные k-расширения графа G.

5. Если на шаге 4 не было найдено ни одного графа, то производится переход на шаг 2.

6. Из всех выбранных вершинных k-расширений на шаге 4 берется по одному представителю от изоморфных графов. Полученное множество представителей являет собой множество попарно-неизоморфных минимальных вершинных k-расширений графа G.

Данный алгоритм является неэффективным и выполняется за экспоненциальное число времени. Подробнее про сложность вычисления вершинных расширений и поиск неизоморфных расширений изложено в работе [1].

Алгоритм 2.4.1 возможно перенести на случай поиска реберных k-расширений естественным путем.

За годы исследований было обнаружено, что алгоритмы, основанные на идее алгоритма 2.4.1 можно ускорить. На помощь снова приходит техника канонических представителей, позволяющая сократить перебор и воспользоваться методикой *isomorphism rejection.* Данную идею использовали и подробно рассмотрели И.А.К. Камил, М.Б. Абросимов, А.А. Лобов для поиска минимальных вершинных расширений [8, 9, 25], а также Х.Х.К. Судани, М.Б. Абросимов, А.А. Лобов – для задачи генерации минимальных реберных расширений [15, 26, 27]. Обе предложенные оптимизации основываются на методе Рида-Фараджева. Результатом их совместной работы стала программная реализация алгоритма поиска всех минимальных вершинных и реберных расширений заданного графа без проверки на изоморфизм, что было реализовано в виде программного комплекса [10].

Приведем теперь задачу поиска минимальных расширений к аналогичной задаче для цветных графов. Базовый алгоритм поиска минимальных вершинных расширений цветных графов выглядит следующим образом.

Алгоритм 2.4.2. Построение всех неизоморфных вершинных k-расширений для заданного i-цветного графа, отличного от вполне несвязного.

Вход: $G = (V, \alpha, f)$, $k \in \mathbb{N}$.

1. $m := 0$.
2. $m := m + 1$.
3. Строятся все графы, полученные из G путем добавления k вершин для каждого цвета $\{1, \dots, i\}$ и m ребер. Общее число вершин в полученных графах будет равно $|V| + ik$.
4. Из полученных на предыдущем шаге графов выбираются все вершинные k-расширения цветного графа G *с учетом цветов*.
5. Если на шаге 4 не было найдено ни одного расширения, то производится переход на шаг 2.
6. Из всех найденных графов выбирается по одному представителю от классов изоморфных графов. Полученное множество представителей является множеством попарно-неизоморфных минимальных вершинных k-расширений цветного графа G.

Аналогично задается алгоритм и для реберных расширений цветных графов с замечанием, что количество вершин в строящихся графах на шаге 3 остается нетронутым.

В алгоритме 2.4.2, помимо вычислительно сложных задач проверки графов на изоморфизм и на то, является ли рассматриваемый граф вершинным расширением заданного, присутствует еще одна нетривиальная проблема – допускает ли рассматриваемый цветной граф вложение заданного цветного графа.

Настоящая работа предлагает воспользоваться методом Рида-Фараджева для построения всех попарно-неизоморфных графов. Данные графы могут быть использованы для вычисления их минимальных расширений. Вычислительные эксперименты, произведенные в ходе главы 1, подтверждают, что метод Рида-Фараджева быстрее метода МакКея для определенных классов

графов. Для решения задачи вложения цветных графов предлагается выбрать особый способ кодирования объектов, который будет рассмотрен далее. Данный способ кодирования и выбора канонических представителей среди набора графов вместе с использованием подхода Рида-Фараджева позволит выработать алгоритм как для построения минимальных вершинных, так и для построения минимальных реберных расширений заданного цветного графа.

В работе предлагается канонический код, состоящий из двух частей: *максимального матричного кода* и *простого кода раскраски*. Подробно рассмотрим обе части кода.

2.5 Максимальный матричный код как часть канонического представителя для расширений цветного графа

Следует сразу оговорить, что все последующие алгоритмы и результаты представлены только для *вершинных раскрасок* графа.

Для ускорения поиска минимальных расширений графа воспользуемся принципами канонических представителей, рассмотренными ранее в работе. В качестве канонического кода представим следующую пару: *максимальный матричный код* и *простой код раскраски* графа. Второй элемент введенной пары позволяет однозначно идентифицировать цветной граф.

В данном параграфе рассмотрим построение *максимального матричного кода* и смысл его использования в решении задачи поиска всех минимальных расширений для заданного цветного графа.

Поскольку в настоящей работе рассматриваются только неориентированные графы, то любой максимальный матричный код будет взят только по верхнему треугольнику матрицы смежности.

Использование максимального матричного кода представляет собой полный инвариант графа, поэтому кодирование графа данным способом позволяет сразу получить представителя от неизоморфных графов с тем же количеством вершин и ребер. Это позволяет сократить время работы алгоритма 2.4.2 на третьем шаге – можно строить графы и их максимальные матричные коды, а далее отсекать те из них, чьи коды совпали. Однако, необязательно сначала строить все графы, а потом исключать из них изоморфные, можно сразу строить неизоморфные. Для этого понадобится особенный способ правильного получения потомка из родителя.

Существует пример того, что не всем алгоритмам подходит использование максимального матричного кода в качестве инварианта из-за некоторых неудобств при генерации одного матричного кода из другого. Допустим, существует некоторый граф, представляющий собой цикл C_5, представленный на рисунке 2.5.1.

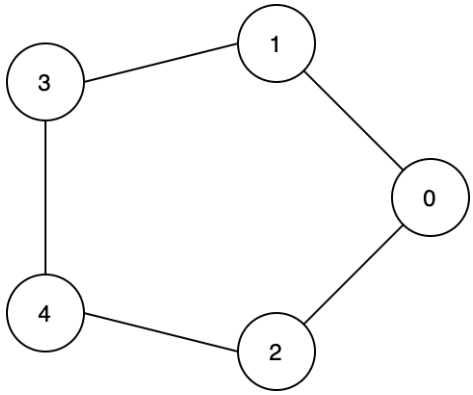

Рисунок 2.5.1 – Циклический граф G_1 с пронумерованными вершинами.

Его максимальным матричный кодом будет следующий (в виде верхнего треугольника матрицы смежности графа):

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

Из цикла C_5 добавлением 1 ребра можно получить только один граф с точностью до изоморфизма. Данный граф изображен на рисунке 2.5.2.

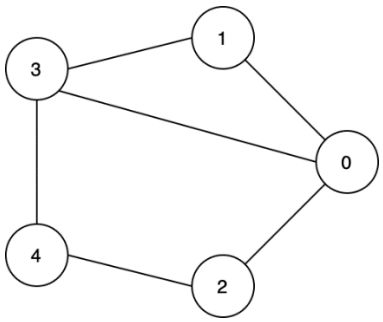

Рисунок 2.5.2 – Граф, полученным путем добавления 1 ребра в C_5.

Следующая матрица смежности представляет матричный код, максимальный среди тех, что могут быть получены из максимального матричного кода G_1 добавлением 1 ребра.

0	1	1	**1**	0
1	0	0	1	0
1	0	0	0	1
0	1	0	0	1
0	0	1	1	0

Однако, данный код не является максимальным, поскольку μ_{max} графа, представленного на рисунке 2.5.2, будет следующим:

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

Итоговый граф с нумерацией, допускающей максимальный матричный код, изображен на рисунке 2.5.3.

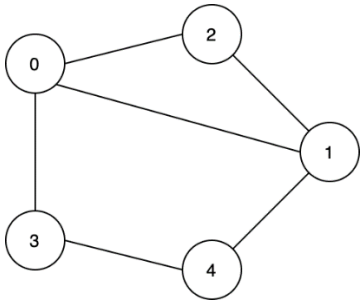

Рисунок 2.5.3 – Рассматриваемый граф C_5 с дополнительным ребром и максимальным матричным кодом.

Именно поэтому следует выбрать другой способ порождения неизоморфных структур, нежели построение по заданному матричному коду.

Сначала рассмотрим само построение максимального матричного кода, поскольку оно будет играть значительную роль при генерации неизоморфных графов без проверки на изоморфизм. Предлагается следующий алгоритм построения максимального матричного кода для заданного неориентированного связного графа. В ходе алгоритма производится вычисление множества всех клик заданного неориентированного графа при помощи алгоритма Брона-Кербоша [54]. Определение клики взято из работы [65].

Определение 2.5.1. Кликой в неориентированном графе $G = (V, \alpha)$ называется подмножество вершин $C \subseteq V$ такое, что для любых двух вершин из C существует ребро, их соединяющее: $\forall\, u, v \in C\colon (u, v) \in \alpha$.

Алгоритм 2.5.1. Построение максимального матричного кода для связного неориентированного графа G.

Вход: $G = (V, \alpha)$.

1. Вычисляется множество всех клик Cq заданного графа;
2. Вычисляется множество вершин D с максимальной степенью $deg(v)$;
3. $max = -1$;
4. **Для всех** $d \in D$:
   - 4.1. $num = 0$;
   - 4.2. $U = \emptyset$ - множество, обозначающее уже использованные вершины;
   - 4.3. $G' = G$;
   - 4.4. $O = \emptyset$, $O \leftarrow d$, где O – список еще не рассмотренных вершин;
   - 4.5. **Пока** $|U| \neq |V|$:
     - 4.5.1. $v \leftarrow O_1$, $O = O \backslash \{v\}$ – первый элемент из списка O помещается в v и удаляется из O.
     - 4.5.2. $Cq(v) = \{cq \in Cq\colon v \in cq\}$ – вычисляется упорядоченное по невозрастанию подмножество клик, содержащих вершину v.
     - 4.5.3. $Cq = Cq \backslash Cq(v)$.

4.5.4. **Для всех** $cq_v \in Cq(v)$ вычисляется, сколько клик из Cq содержит каждый элемент из cq_v. Элементы сортируются по убыванию количества клик. Элементы с разным количеством клик сортируются по степени вершин deg.

4.5.5. Для каждого отсортированного cq_v **для всех** $a \in cq_v$: $a \notin U$ очередная вершина a перенумеровывается в графе G' на значение num и num увеличивается на единицу.

4.5.6. Для каждого соседа u вершины v, где $u \notin U$, вычисляется, сколько клик из Cq содержит u.

4.5.7. Множество $nb(v)$ смежных с v вершин сортируется по невозрастанию количества клик, полученных на предыдущем шаге. Соседи с равным количеством клик сортируются по степени deg.

4.5.8. **Для всех** $b \in nb(v)$: $b \notin U$ каждая вершина b перенумеровывается в графе G' на значение num, заносится в список O: $O \leftarrow b$, и далее num увеличивается на единицу.

4.5.9. В конце очередной итерации список O упорядочивается в порядке перенумерованных вершин.

4.6. Берется верхний треугольник матрицы смежности графа G': $rut(G')$ и, если он больше текущего max ($rut(G') > max$), то max обновляется в соответствии со значением $rut(G')$: $max = rut(G')$.

5. Выводится max – максимальный матричный код связного графа G.

Общая идея алгоритма 2.5.1 заключается в поиске максимальных клик, содержащих вершины с максимальной степенью и последовательное рассмотрение максимальных клик, включающих смежные вершин и так далее. При этом на каждом шаге следования производится перенумерация вершин с наибольшей *на данный момент* степенью, обеспечивая граф наибольшим числом значащих единиц в максимальном матричном коде.

Стоит заметить, что алгоритм 2.5.1 может быть расширен на случай несвязного неориентированного графа с несколькими компонентами связности (вплоть до вполне-несвязного графа O_n). Для этого необходимо:

1. Вычислить максимальный матричный код каждого компонента связности.

2. Упорядочить коды по невозрастанию.

3. Расположить полученные коды на верхнем правом треугольнике результирующей матрицы смежности таким образом, чтобы каждый следующий код начинался на следующей строке и следующем столбце после последнего элемента предыдущего кода.

Последний пункт означает, что если предыдущий код в результирующей матрице закончился на строке i и столбце j, то следующий код будет добавлен на $i + 1$ строку и $j + 1$ столбец.

Теперь, когда мы обладаем способом строить максимальные матричные коды, необходимо задать канонический способ генерировать потомков из родителя. Для этого на помощь приходят несколько важных правил при генерации новых кандидатов на расширение путем добавления одного дополнительного ребра к текущему графу.

Перед тем, как задать алгоритм каноничного построения потомков из родителя, зададим функцию $Orb(G)[v]$, которая равна множеству подобных вершин, в котором содержится вершина v.

Алгоритм 2.5.2. Построение максимальных матричных кодов всех неизоморфных графов, получающихся из графа G' путем добавления одного дополнительного ребра.

Вход: $G' = (V', \alpha')$ или $G' = (V', \alpha', f')$; $\mu_{max}(G')$.

1. Находим множество орбит графа $Orb(G')$ (в соответствии с разбиением, если граф цветной).

2. Строим множество представителей P_{orb} из множества орбит, забирая из каждой по одному представителю.

3. Для каждой вершины v определяем множество Alw_v – в какие вершины «разрешено» проводить дополнительные ребра по следующим критериям:
    a. ребро «разрешено» проводить в любые вершины, которые содержатся в орбите вершины v.
    b. вершина не из орбиты вершины v попадает в множество разрешенных, когда она содержится в множестве представителей: $u \in Alw_v \Leftrightarrow u \in P_{orb}$, и ее номер больше номера вершины v.
4. Для каждой вершины из множества представителей $u \in P_{orb}$ строятся новые графы путем добавления дополнительного ребра в $\mu_{max}(G')$ по следующим критериям:
    a. $u \in Alw_v$;
    b. $v < u$ – номер вершины u больше номера v;
    c. $(v, u) \notin \alpha'$.

В ходе вычислительных экспериментов, использующих реализацию алгоритма 2.5.2 на различных классах графов, было обнаружено, что на шаге 2 эффективнее выбирать первого по номеру представителя, а на шаге 4 эффективнее рассматривать вершины u от большей по номеру к меньшей.

2.6 Простой код раскраски как часть канонического представителя для расширений цветного графа

Вторая часть канонического представителя является специальным способом нумерации цветного графа для проверки вложения двух цветных графов с учетом цветов. Данный способ назовем *кодом раскраски.*

Элемент кода раскраски предлагается строить следующим образом:

1. Рассматривается вершина $v \in V$, записывается ее цвет.

2. Рассматриваются все ее смежные вершины, и подсчитывается количество смежных вершин каждого цвета.

Подсчитанные данные записываются в следующем формате для каждой вершины v:

$$f(v)\{|f_1|, |f_2|, \dots, |f_j|\},$$

где $f(v)$ – цвет вершины v, а $|f_i|$ – количество смежных вершине v вершин цвета f_i.

Например, данная структура для некой вершины 2-цветного графа может быть следующей: $1(3,1)$. Эта структура означает, что вершина цвета 1 смежна трем вершинам цвета 0 и одной – цвета 2.

Вектор структур для всех вершин образует *код раскраски*. К сожалению, подобным кодом довольно сложно пользоваться при сравнении и поиска ответа на вопрос, включает ли один код раскраски другой. Сравнение двух векторов подразумевает проход по двух векторам и является процедурой с вычислительной сложностью $O(n^2)$. Поэтому предлагается модифицировать код раскраски таким образом, чтобы минимизировать количество операций сравнения.

Модификация кода раскраски происходит следующим образом. Для каждого цвета f_i выбирается два простых числа: f_i^1 и f_i^2. Первое число стоит брать небольшим, например 3, 7, 11 и так далее. Второе следует задавать больше, чем первое в степени количества вершин графа: $f_i^2 > (f_i^1)^n$. Тогда для каждой структуры $f(v)\{|f_1|, |f_2|, \dots, |f_j|\}$, построенной для некоторой вершины $v \in V$ графа G, вычисляется число, которое называется *простым кодом вершины:*

$$fc(v) = f_i^2 \cdot (f_1^1)^{|f_1|} \cdot (f_2^1)^{|f_2|} \cdot \dots \cdot \left(f_j^1\right)^{|f_j|}.$$

Легко заметить, что данное число является произведением простых чисел. Вектор простых кодов всех вершин образует *простой код раскраски*. Он обозначается следующим образом:

$$FC(G) = (fc(v): v \in V)$$

Используя данные простые коды раскрасок, проверка, допускает ли цветной граф G_1 вложение цветного графа G_2, сводится к следующему алгоритму.

Зададим функцию $FC(G)[v] = fc(v)$, где $G = (V, \alpha, f)$ и $v \in V$.

Алгоритм 2.6.1. Проверка вложенности графа G_2 в граф G_1.

Вход: $G_1 = (V_1, \alpha_1, f_1)$, $G_2 = (V_2, \alpha_2, f_2)$.

1. $i = 0, j = 0$.
2. Строятся $FC(G_1)$ и $FC(G_2)$.
3. **Пока** $i \leq |FC(G_1)|$:
   - 3.1. **Если** $j > |FC(G_2)|$, то граф G_2 вкладывается в граф G_1. Выход из алгоритма.
   - 3.2. **Если** $FC(G_1)[i]\ mod\ FC(G_2)[j] = 0$, то $j = j + 1$.
   - 3.3. $i = i + 1$.
4. **Если** $j > |FC(G_2)|$, то граф G_2 вкладывается в граф G_2. Выход из алгоритма.
5. Иначе граф G_2 не может быть вложен в граф G_1.

Данный алгоритм дает понимание не только о том, что граф может быть вложен в другой граф, но и гарантирует сохранность цветов при вложении.

2.7 Использование метода Рида-Фараджева для поиска минимальных вершинных расширений цветных графов

Смысл использования метода канонических представителей заключается в следующем. Рассмотрим общий ход алгоритма 2.4.2. В начале алгоритма предлагается построить максимальный матричный код и простой код раскраски для исходного графа: $\left(\mu_{max}(G), FC(G)\right)$. На третьем шаге мы строим все графы путем добавления в исходный граф m ребер и, если целью поиска являются вершинные расширения, ik дополнительных вершин. На данном шаге следует строить графы совместно с описанными выше парами $\left(\mu_{max}\left(G_j^m\right), FC\left(G_j^m\right)\right)$, где G_j^m – некоторый граф, построенный на шаге 3. Очевидно, графы с одинаковыми парами смысла рассматривать далее не имеет, поскольку эти графы изоморфны. Оставляя после шага 3 только те графы, у которых пары не совпадают, мы получаем множество попарно-неизоморфных графов с $\left|\alpha^j\right| = |\alpha| + m$ и, если производится поиск вершинных раскрасок, $\left|V^j\right| = |V| + ik$.

Проверка каждого полученного графа на то, является ли он расширением заданного графа происходит следующим образом. Перебираются все возможные k вершин (ребер) и с учетом каждого такого набора строится простой код раскраски очередного графа G_j^m. При построении данного кода раскраски текущий набор из k вершин (ребер) в нее не включается. Далее каждый полученный таким образом простой код раскраски сравнивается с кодом исходного графа: если код исходного графа является каноническим родителем проверяемого, то считается, что текущий граф является частичным расширением. Частичное расширение означает, что, если произойдет отказ именно тех вершин (ребер), которые в данный момент находятся в рассмотрении, то для заданного цветного графа существует вложение в текущий. Если же текущий

граф при любом наборе из k вершин (ребер) допускает вложение исходного графа, то он объявляется минимальным вершинным (реберным) k-расширением заданного графа.

Представим алгоритм по описанной выше идее. Данный алгоритм содержит вызов алгоритмов 2.5.1, 2.5.2, 2.6.1 и в результате выводит множество попарно-неизоморфных минимальных вершинных k-расширений.

Алгоритм 2.7.1. Построение всех неизоморфных минимальных вершинных k-расширений неориентированного i-цветного n-вершинного графа.

Вход: $G = (V, \alpha, f)$, $|V| = n$; $f: V \to F$, $F = \{1, 2, \dots, i\}$, $|F| = i$.

1. $C = \emptyset$ – множество кандидатов на текущем шаге.

2. $E = \emptyset$ – результирующее множество минимальных вершинных k-раскрасок графа G.

3. Вычисляется простой код раскраски графа G и сохраняется в $FC_{org} = FC(G)$.

4. Строится граф $G' = (V', \alpha', f')$, где множество вершин нового графа $V' = V \cup \{n+1, n+2, \dots, n+ik\}$, множество ребер $\alpha' = \alpha$, а функция раскраски действует на все вершины нового графа $f': V' \to \{1, 2, \dots, i\}$. Обозначим вектор раскраски графа G' как $f'(G')$.

5. Строятся все возможные перестановки из k вершин графа G' и записывается в множество P.

6. Вычисляется максимальный матричный код $\mu_{max}(G')$ графа G' при помощи алгоритма 2.5.1.

7. $C \leftarrow \{\mu_{max}(G')\}$ – построенный граф добавляется в множество кандидатов как одноэлементное множество.

8. **Пока** $|C| > 0$:

8.1. Если $E \neq \emptyset$, то выводится E как множество попарно-неизоморфных минимальных вершинных k-раскрасок и алгоритм завершается.

8.2. Вырабатывается множество новых попарно-неизоморфных кандидатов C_{new} при помощи алгоритма 2.5.2, последовательно подавая ему

на вход $\mu_{max}(C_r)$ и $f'(G')$ – раскраску графа G', которая, очевидно, является также раскраской каждого графа из C. Таким образом, если множество C содержало несколько элементов, то C_{new} будет являться объединением выработанных множеств для каждого из этих элементов.

8.3. Если $|C_{new}| = 0$, то переход на шаг 8.

8.4. **Для всех** $G_j \in C_{new}$:

8.4.1. Для каждой $p \in P$ из графа G_j строится простой код раскраски FC_j, *не включающий вершины из* p.

8.4.2. Для каждого простого кода раскраски FC_j проверяется, допускает ли код FC_j вложение кода FC_{org} при помощи введенного алгоритма 2.6.1. Если какой-либо FC_j не допускает вложения FC_{org}, то соответствующий рассматриваемому коду граф G_j отсекается, поскольку не является вершинным расширением графа G.

8.4.3. Если все FC_j допускают вложение FC_{org}, то граф G_j объявляется вершинным расширением и записывается в множество расширений $E \leftarrow G_j$.

8.5. $C = C_{new}$ и переход на шаг 8.

9. Выводится $E = \emptyset$, то есть объявляется, что минимальных вершинных k-расширений для графа G' не найдено.

Помимо описанных способов отсечения неканонических представителей с целью сокращения перебора в ходе реализации алгоритма 2.7.1 были задействованы несколько других оптимизаций.

Во-первых, несложно заметить, что с целью упрощения работы алгоритма, необходимые ik вершин по второму условию определения минимальных вершинных расширений добавляются в исходный граф сразу. Таким образом, все дальнейшие операции генерации заключаются в подборе правильного расположения ребер на графе. Кроме того, можно сразу посчитать все возможные комбинации отказа k вершин, что и происходит на шаге 5.

Во-вторых, внутри цикла по всем G_j из C_{new} первым шагом можно произвести проверку, допускает ли цветной граф G_j вложение исходного графа G. Если вложения нет, то далее рассматривать G_j не имеет смысла и можно переходить к следующему графу G_{j+1}.

В-третьих, подобный алгоритм легко может быть реализован в многопоточном режиме, что и сделано в программном комплексе [37]. Сюда подходит схема *producer-consumer*, где поток типа *producer* вырабатывает новых кандидатов на проверку C_{new}, а несколько потоков типа *consumer* проверяют их на условие вершинного расширения.

Для удобства алгоритм 2.7.1 будем называть **Г-МВР** (**Г**енерация **М**инимальных **В**ершинных **Р**асширений).

2.8 Вычислительные эксперименты генерации неизоморфных минимальных вершинных расширений для заданных цветных графов

Алгоритм Г-МВР был реализован и включен в программный комплекс [37]. Были проведены вычислительные эксперименты для неориентированных графов различного количества вершин. Кроме того, отдельными экспериментами были получены результаты на двух наиболее интересующих нас классах графов: полные графы и графы-звезды.

Общая схема тестирования выглядит следующим образом:

1. Берется список неизоморфных графов заданного класса с количеством вершин от 2 до 11.

2. Для этого списка вычисляются множества неизоморфных цветных графов при помощи алгоритма Г-РФ. В результате получается такая матрица:

для каждого числа вершин n и каждой возможной раскраски получается некоторое множество попарно-неизоморфных n-вершинных графов, которые допускают рассматриваемую раскраску.

3. Для каждого полученного цветного графа запускается алгоритм Г-МВР.

Список всех попарно-неизоморфных графов заданного числа вершин может быть найден в базе данных Брендана МакКея [90] в формате *graph6* [89].

Здесь используется алгоритм генерации цветных графов Г-РФ, поскольку он показал более эффективные результаты при генерации неизоморфных раскрасок.

Поскольку после обработки двумя алгоритмами Г-РФ и Г-МВР одного графа получается достаточно объемный вывод, приведем здесь избранные примеры вычислений – приведем результаты построения МВ-1Р графов от 6 до 8 вершин с количеством цветов до трех. Результаты представлены в виде числа графов, соответствующих некоторой раскраске, полученных после работы алгоритма Г-РФ. Для данного количества записывается время поиска вершинного расширения алгоритмом Г-МВР для одного и для всех графов.

Мощность множеств вершин определенного цвета будем обозначать буквами $a = |f_1|$, $b = |f_2|$, $c = |f_3|$. То есть, если некоторый граф содержит 1 вершину первого цвета f_1 и 5 вершин второго f_2 цвета, значит $a = 1, b = 5$. Данное обозначение содержит в себе множество раскрасок, поскольку $a = 1, b = 5$ может быть сопоставлено следующим векторам раскрасок:

$$(0,1,1,1,1,1),(1,0,1,1,1,1),(1,1,0,1,1,1),$$
$$(1,1,1,0,1,1),(1,1,1,1,0,1),(1,1,1,1,1,0)$$

Для примера таблица 2.8.1 отражает количество неизоморфных графов, сопоставленных с векторами двухцветных раскрасок, представленными выше.

Таблица 2.8.1 – Общее число попарно-неизоморфных шестивершинных цветных графов, соответствующих векторам раскраски при $a = 1, b = 5$

Вектор раскраски с $a = 1, b = 5$	Количество неизоморфных графов
$(0, 1, 1, 1, 1, 1)$	155
$(1, 0, 1, 1, 1, 1)$	69
$(1, 1, 0, 1, 1, 1)$	61
$(1, 1, 1, 0, 1, 1)$	44
$(1, 1, 1, 1, 0, 1)$	50
$(1, 1, 1, 1, 1, 0)$	66

Такие количества обуславливаются тем, что некоторый граф с раскраской $(1, 0, 1, 1, 1, 1)$ может быть изоморфен либо совпадать с некоторым графом с раскраской $(1, 1, 1, 1, 1, 0)$. Поэтому, в результирующую таблицу будет записано суммарное количество попарно-неизоморфных цветных графов. Так, для таблицы 2.8.1 суммарное количество графов будет равно $155 + 69 + 61 + 44 + 50 + 66 = 445$.

Итак, когда все необходимые обозначения введены, представим результаты вычислительных экспериментов.

На таблице 2.8.2 представлены результаты по поиску MB-1P для графов с количеством вершин, равным шести, и с количеством цветов 2 и 3 соответственно.

Таблица 2.8.2 - Результаты вычислительных экспериментов по поиску MB-1P для всех попарно-неизоморфных графов с шестью вершинами

Раскраска	Количество неизоморфных цветных графов	Среднее время работы Г-МВР для одного графа	Общее время работы Г-МВР
$a = 1, b = 5$	445	316 мс.	5 мин. 39 сек. 218 мс.
$a = 2, b = 4$	843	289 мс.	11 мин. 24 сек. 761 мс.
$a = 3, b = 3$	1019	281 мс.	11 мин. 13 сек. 445 мс.
$a = 1, b = 1, c = 4$	1561	331 мс.	11 м. 15 с. 344 мс.
$a = 1, b = 2, c = 3$	2843	320 мс.	17 м. 29 с. 326 мс.
$a = 2, b = 2, c = 2$	4059	292 мс.	20 м. 17 с. 305 мс.

Следующие результаты, представленные на таблице 2.8.3, были получены для всех неизоморфных 7-вершинных графов. Для них были построены МВ-1Р, общее и среднее время поиска которых и отражено в таблице.

Таблица 2.8.3 - Результаты вычислительных экспериментов по поиску МВ-1Р для всех попарно-неизоморфных графов с семью вершинами

Раскраска	Количество неизоморфных цветных графов	Среднее время работы Г-МВР для одного графа	Общее время работы Г-МВР
$a = 1, b = 6$	3594	295 мс.	18 м. 10 с. 302 мс.
$a = 2, b = 5$	7736	284 мс.	37 м. 47 с. 288 мс.
$a = 3, b = 4$	11412	274 мс.	1 ч. 37 с. 284 мс.
$a = 1, b = 1, c = 5$	15628	347 мс.	2 ч. 40 м. 26 с. 352 мс.
$a = 1, b = 2, c = 4$	35314	314 мс.	4 ч. 26 м. 52 с. 329 мс.
$a = 1, b = 3, c = 3$	45782	301 мс.	5 ч. 17 м. 24 с. 302 мс.
$a = 2, b = 2, c = 3$	67030	292 мс.	7 ч. 27 м. 11 с. 307 мс.

Аналогично были получены результаты по поиску минимальных вершинных 1-расширений для всех графов с восемью вершинами. Результаты вычислительных экспериментов содержатся в таблице 2.8.4.

Таблица 2.8.4 - Результаты вычислительных экспериментов по поиску МВ-1Р для всех попарно-неизоморфных графов с восемью вершинами

Раскраска	Количество неизоморфных цветных графов	Среднее время работы Г-МВР для одного графа	Общее время работы Г-МВР
$a = 1, b = 7$	48543	290 мс.	6 ч. 56 м. 6 с. 297 мс.
$a = 2, b = 6$	115527	271 мс.	16 ч. 42 м. 29 с. 280 мс.
$a = 3, b = 5$	192825	266 мс.	28 ч. 18 м. 3 с. 274 мс.
$a = 4, b = 4$	221573	247 мс.	29 ч. 19 м. 24 с. 261 мс.
$a = 1, b = 1, c = 6$	254087	366 мс.	50 ч. 53 м. 37 с. 378 мс.
$a = 1, b = 2, c = 5$	677057	328 мс.	60 ч. 45 м. 30 с. 342 мс.
$a = 1, b = 3, c = 4$	1074909	303 мс.	88 ч. 30 м. 20 с. 304 мс.
$a = 2, b = 2, c = 4$	1594165	297 мс.	130 ч. 41 м. 59 с. 311 мс.
$a = 2, b = 3, c = 3$	2127843	280 мс.	162 ч. 31 м. 50 с. 281 мс.

В качестве отдельного эксперимента найдем МВ-kР для полных и для звездных цветных графов некоторого количества вершин $k \geq 1$ и $i \geq 2$. Это позволит нам проанализировать полученный результат и, возможно, найти некоторые закономерности в схеме построения. Данные закономерности обычно позволяют найти общую схему построения минимальных расширений для заданного графа. Подобные эксперименты уже были ранее поставлены для некоторых классов цветных графов, таких как циклы и цепи. Для этих экспериментов было написано программное решение, запатентованное Роспатентом [4]. В этой работе будут исследоваться полные графы и звезды на существование некоторой общей схемы построения.

Начнем с полных графов. Рассмотрим результаты экспериментов для всех цветных полных графов до семи вершин. Были рассмотрены двух- и трехцветные раскраски, а степень расширения была взята от 1 до 3.

Таблица 2.8.5 представляет результаты запуска программы и содержит среднее время обработки одного цветного графа и общее время работы для всех цветных полных графов заданного количества вершин. В таблице для графов общего вида $G = (V, \alpha, f)$ параметр $n = |V|$, k – степень расширения, раскраска представлена в виде обозначения, описанного выше. Для обозначения минимальных вершинных k-расширений будем пользоваться сокращением МВР.

По результатам данных запусков была собрана база МВР цветных полных графов. Этой база позволяет удостовериться, что теоретические рассуждения и аналитические схемы построения минимальных расширений для цветных полных графов совпадают с полученными в ходе вычислительных экспериментов практическими результатами.

Таблица 2.8.5 – Результаты поиска всех неизоморфных МВР для цветных полных графов

n	Количество неизоморфных цветных графов	k	Общее время работы Г-МВР	Среднее время работы Г-МВР для одного графа
2	1	1	15 мс.	15 мс.
		2	19 мс.	19 мс.
3	3	1	190 мс.	61 мс.
		2	235 мс.	74 мс.
		3	279 мс.	89 мс.
4	6	1	1 сек. 159 мс.	184 мс.
		2	1 сек. 345 мс.	206 мс.
		3	1 сек. 587 мс.	232 мс.
5	19	1	4 сек. 685 мс.	245 мс.
		2	5 сек. 134 мс.	261 мс.
		3	5 сек. 302 мс.	279 мс.
6	94	1	30 сек. 139 мс.	310 мс.
		2	31 сек. 942 мс.	335 мс.
		3	35 сек. 392 мс.	368 мс.
7	635	1	5 мин. 13 сек. 893 мс.	422 мс.
		2	5 мин. 42 сек. 345 мс.	465 мс.
		3	6 мин. 58 сек. 945 мс.	496 мс.

Звезды стали еще одним классом графов, которые были поданы на вход экспериментальных запусков. В данной работе предлагаются результаты данных экспериментов для всех цветных звездных графов с количеством вершин до семи. Были рассмотрены 2- и 3-цветные раскраски, а степень расширения, как и в предыдущем эксперименте, была рассмотрена от 1 до 3.

Таблица 2.8.6 содержит результаты запуска программы для графов-звезд и содержит среднее время обработки одного цветного графа и общее время работы для всех цветных полных графов заданного количества вершин. В таблице для графов общего вида $G = (V, \alpha, f)$ параметр $n = |V|$, k – степень расширения, раскраска представлена в виде обозначения, описанного выше.

Таблица 2.8.6 – Результаты поиска всех неизоморфных МВР для цветных звездных графов

n	Количество неизоморфных цветных графов	k	Общее время работы Г-МВР	Среднее время работы Г-МВР для одного графа
2	1	1	10 мс.	10 мс.
		2	12 мс.	12 мс.
3	3	1	170 мс.	53 мс.
		2	183 мс.	55 мс.
		3	200 мс.	60 мс.
4	6	1	583 мс.	87 мс.
		2	593 мс.	91 мс.
		3	711 мс.	113 мс.
5	19	1	2 сек. 333 мс.	121 мс.
		2	2 сек. 580 мс.	134 мс.
		3	3 сек. 19 мс.	156 мс.
6	94	1	19 сек. 412 мс.	205 мс.
		2	21 сек. 8 мс.	219 мс.
		3	23 сек. 223 мс.	247 мс.
7	635	1	4 мин. 48 сек. 395 мс.	344 мс.
		2	5 мин. 33 сек. 284 мс.	361 мс.
		3	7 мин. 12 сек. 948 мс.	389 мс.

Как и для цветных полных графов, для цветных звезд была собрана база графов. Данная база позволяет сравнить графы, построенные по аналитическим схемам построения, с графами, полученными в ходе вычислительных экспериментов.

Отдельным случаем были рассмотрены минимальные реберные k-расширения для цветных звезд. Для этого был модифицирован алгоритм Г-МВР таким образом, чтобы при проверке вложения с учетом цветов из графа удалялись k ребер. Как и в предыдущем эксперименте, рассматривались звезды до

7 вершин с $i = 2$ и $i = 3$ и $k = \overline{1,\ldots,3}$. Таблица 2.8.7 содержит результаты запуска с описанными входными данными. Минимальные реберные k-расширения будем обозначать сокращением МРР.

Таблица 2.8.7 – Результаты поиска всех неизоморфных МРР для цветных звездных графов

n	Количество неизоморфных цветных графов	k	Общее время работы алгоритма поиска МРР	Среднее время работы алгоритма поиска МРР для одного графа
2	1	1	6 мс.	6 мс.
		2	7 мс.	7 мс.
3	3	1	56 мс.	15 мс.
		2	77 мс.	18 мс.
		3	80 мс.	22 мс.
4	6	1	324 мс.	53 мс.
		2	387 мс.	60 мс.
		3	415 мс.	69 мс.
5	19	1	1 сек. 976 мс.	100 мс.
		2	2 сек. 169 мс.	113 мс.
		3	2 сек. 534 мс.	119 мс.
6	94	1	12 сек. 501 мс.	132 мс.
		2	13 сек. 93 мс.	145 мс.
		3	14 сек. 720 мс.	156 мс.
7	635	1	1 мин. 58 сек. 568 мс.	289 мс.
		2	4 мин. 34 сек. 109 мс.	311 мс.
		3	5 мин. 15 сек. 684 мс.	334 мс.

На основании полученных данных была собрана база минимальных реберных расширений для цветных звезд.

Глава 3. Минимальные расширения цветных полных графов

В данной главе используется аналитический подход для поиска некой общей схемы минимальных вершинных расширений для цветных полных графов. Очевидно, что не существует реберных расширений полных графов, поэтому рассматривались только минимальные вершинные расширения. Глава содержит последовательные рассуждения о схемах построения минимальных расширений, каждый шаг рассуждений обобщает предыдущие полученные и доказанные результаты. Вычислительные эксперименты по запуску реализации алгоритма Г-МВР на классе цветных полных графов позволили сравнить полученные аналитическим путем результаты с полученными практическим путем. Результаты совпали.

В главе используются обозначения и сокращения, призванные облегчить формулирование схем построения минимальных расширений для цветных графов.

Пусть есть граф $G = (V, \alpha, f)$. Множество $V_{f_j} = \{v \mid f(v) = f_j\}$ – множество вершин из V, которым сопоставлен цвет f_j. Очевидно, что $V_{f_j} \subset V$. Тогда множество, содержащее все множества набора вершин по цветам будем обозначать за $W = \left\{V_{f_j} \middle| \forall f_j \in F\right\}$.

Множество $W^1 = \left\{V_{f_j} : V_{f_j} \in W, |V_{f_j}| = 1\right\}$ содержит все множества вершин с цветом, встречающимся лишь единожды в графе. Множество, которое содержит наборы вершин, сопоставленных с цветами, встречающимися в графе более одного раза, обозначается за $W^* = \left\{V_{f_j} : V_{f_j} \in W, |V_{f_j}| > 1\right\}$.

На всех рисунках, содержащих расширение графа $G^* = (V^*, \alpha^*, f^*)$, ребра, обозначенные пунктирной линией, являются ребрами расширения из

$\alpha^*\backslash\alpha$, или *дополнительными ребрами.* Вершины же, обозначенные пунктирным контуром, являются вершинами расширения: $v \in V^*\backslash V$. Данные вершины будем называть *дополнительными вершинами.*

Исходные вершины v из V цвета f_j будем обозначать v_{f_j}. Тогда дополнительные вершины u из $V^*\backslash V$ цвета f_j будут обозначены за $u^+_{f_j}$. Аналогично введем обозначения вершин, включенных в множества из W^1 за v_{W^1} и $v^+_{W^1}$ для исходных и дополнительных соответственно. Аналогично введем v_{W^*} и $v^+_{W^*}$ для W^*.

Введем функцию $F(W)$ – это уникальный набор цветов, сопоставленных вершинам из множеств, включенных в множество W. Тогда значением функции $F(W^1)$ будет являться набор всех цветов, встречающихся в графе единожды, а значением $F(W^*)$ – множество всех неуникальных цветов в графе.

Схемы построения минимальных вершинных k-расширений цветных полных графов будем рассматривать от частного случая к общему, постепенно обобщая частные случаи до тех пор, пока не получим общую схему построения МВР для цветных полных графов.

3.1 Минимальные вершинные 1-расширения двухцветных полных графов

Начнем с МВ-1Р двухцветных полных графов. Другими словами, условиями данного параграфа будут следующие значения i и k: $i = 2, k = 1$. Тогда $F = \{f_1, f_2\}$.

При заданных условиях можно выделить два случая цветных полных графов:

– $|V_{f_1}| = 1$, $|V_{f_2}| > 1$ – в графе одна вершина сопоставляется цвету f_1, а остальные – цвету f_2.

– $|V_{f_1}| > 1$, $|V_{f_2}| > 1$ – в графе есть несколько вершин цвета f_1 и несколько – цвета f_2.

Кроме того, существует единственный с точностью до изоморфизма граф, удовлетворяющий условиям $|V_{f_1}| = 1$, $|V_{f_2}| = 1$ – это двухвершинных полный граф K_2. Схема построения для подобных графов будет рассмотрена в обобщенном виде.

Теорема 3.1.1. Полный граф $K_n = (V, \alpha, f)$ с $F = \{f_1, f_2\}$, $|F| = 2$, $|V_{f_1}| = 1$, $|V_{f_2}| > 1$ имеет единственное с точностью до изоморфизма минимальное вершинное 1-расширение с числом дополнительных ребер, равным $2n - 1$.

Схема построения минимального вершинного 1-расширения:

1. Из дополнительной вершины цвета f_1 проводятся ребра во все исходные вершины цвета f_2.

2. Из дополнительной вершины цвета f_2 проводятся ребра во все исходные вершины.

Доказательство. Рассмотрим оба пункта схемы по отдельности.

Для наглядности будем иллюстрировать шаги доказательства на примере K_4, удовлетворяющего условиям теоремы. Пример подобного цветного полного графа представлен на рисунке 3.1.1. Здесь цвет f_1 обозначается синим цветом, а f_2 – белым.

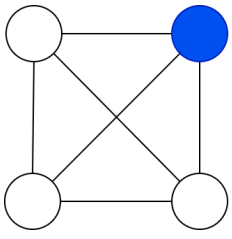

Рисунок 3.1.1 – Пример двухцветного полного графа K_4.

При удалении исходной вершины цвета f_1 все исходные вершины цвета f_2 теряют свою связь с вершинами цвета f_1. Таким образом, исходный граф

становится одноцветным. Для того, чтобы восстановить исходный граф, необходимо соединить ребрами дополнительную вершину цвета f_1 со всеми исходными вершинами цвета f_2. Количество дополнительных ребер в данном случае будет равным $n-1$. Пример удаления вершины цвета f_1 и восстановление цветного вложения исходного графа представлено на рисунке 3.1.2.

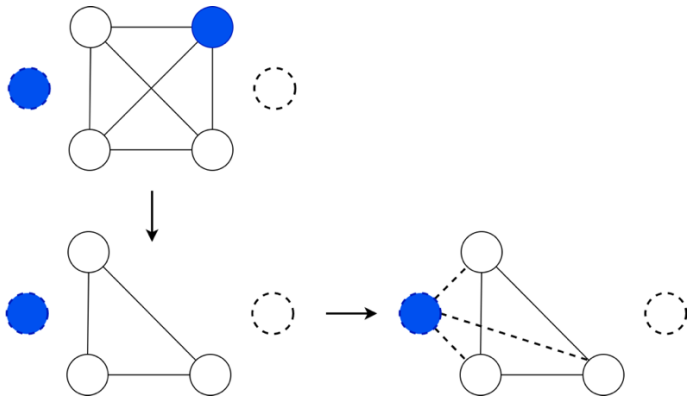

Рисунок 3.1.2 – Удаление вершины цвета f_1 и восстановление вложенности соединением дополнительной вершины цвета f_1 с исходными.

При удалении исходной вершины цвета f_2 граф превращается в полный двухцветный граф K_{n-1}. Чтобы вернуть возможность вложить исходный граф в расширение, необходимо соединить дополнительную вершину цвета f_2 со всеми исходными вершинами со степенью $n-1$ (вершины, потерявшие ребро после отказа рассматриваемой вершины). Пример описанных действий изображен на рисунке 3.1.3.

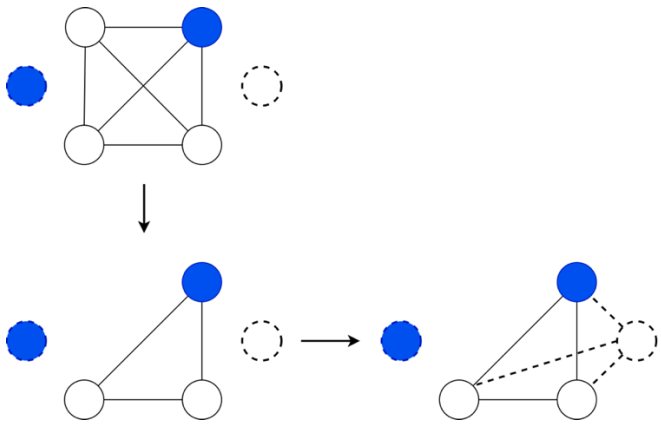

Рисунок 3.1.3 – Пример удаления исходной вершины цвета f_2 и восстановления утраченных ребер.

С учетом условия вершинного расширения, при котором может быть удалена любая вершина f_2, ребра будут проведены ко всем исходным вершинам. Их общее количество будет равно n. Все возможные варианты отказа вершин цвета f_2 для полного графа K_4 изображены на рисунке 3.1.4.

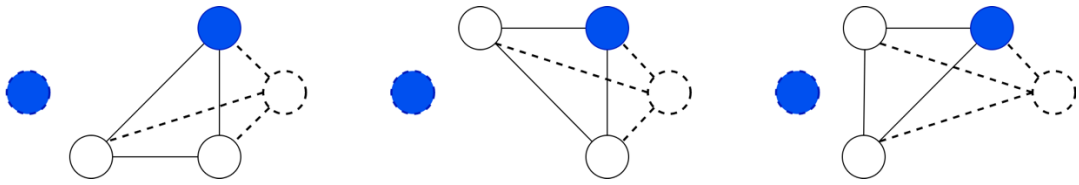

Рисунок 3.1.4 – Все возможные варианты отказа вершины цвета f_2 в цветном полном графе K_4.

Итоговая сумма дополнительных ребер будет $n-1+n=2n-1$. ■

Пример минимального вершинного расширения цветного графа K_4, построенного по доказанной схеме, представлен на рисунке 3.1.5.

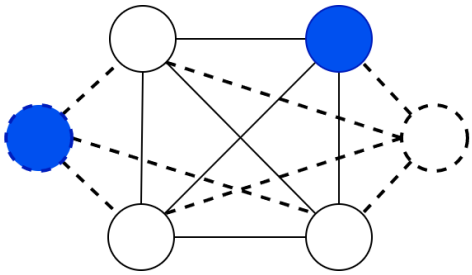

Рисунок 3.1.5 – Пример МВ-1Р цветного графа K_4 из примера в доказательстве.

Теорема 3.1.2. Полный граф $K_n=(V,\alpha,f)$ с $F=\{f_1,f_2\}$, $|F|=2$, $|V_{f_1}|>1$, $|V_{f_2}|>1$ имеет единственное с точностью до изоморфизма минимальное вершинное 1-расширение с количеством дополнительных ребер, равным $2n$.

Схема построения минимального вершинного 1-расширения:

1. Из дополнительной вершины цвета f_1 проводятся ребра во все исходные вершины.

2. Из дополнительной вершины цвета f_2 проводятся ребра во все исходные вершины.

Доказательство. Аналогично предыдущей теореме, при удалении вершины цвета f_1 $n-1$ исходных вершин теряют связь с данной вершиной, поэтому необходимо добавить $n-1$ ребер между исходными вершинами и дополнительной вершиной цвета f_1. Поскольку отказать может любая вершина

цвета f_1, то дополнительная вершина цвета f_1 должна быть соединена со всеми исходными вершинами. При этом количество дополнительных ребер будет равно n.

Аналогичным образом дополнительная вершина цвета f_2 должна быть соединена со всеми исходными вершинами.

Итоговое количество дополнительных ребер будет равно $n + n = 2n$. ■

Граф K_n, удовлетворяющий условиям теоремы, и его минимальное вершинное 1-расширение представлены на рисунке 3.1.6.

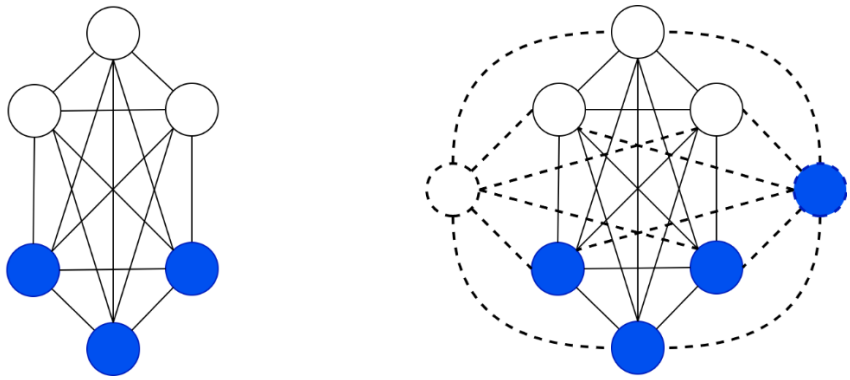

Рисунок 3.1.6 – Пример двухцветного полного графа и его МВ-1Р, построенного по схеме из теоремы 3.1.2.

3.2 Минимальные вершинные k-расширения двухцветных полных графов

Перейдем к рассмотрению МВР, у которых $k > 1$, для двухцветных полных графов, таким образом обобщая случай построения МВ-1Р двухцветных полных графов. Представленные в этом параграфе схемы дадут представление о том, как должны строиться минимальные k-отказоустойчивые реализации цветных полных графов.

Теорема 3.2.1. Полный граф $K_n = (V, \alpha, f)$ с $F = \{f_1, f_2\}$, $|F| = 2$, $\left|V_{f_1}\right| = 1$, $\left|V_{f_2}\right| > 1$ имеет единственное с точностью до изоморфизма минимальное вершинное k-расширение с количеством дополнительных ребер, равным:

$$k(n-1) + kn + (k-1)^2 + \frac{k(k-1)}{2}$$

Минимальное вершинное k-расширение строится по следующей схеме:

1. Из k дополнительных вершин цвета f_1 проводятся ребра во всех исходные вершины цвета f_2. Их количество равно $k(n-1)$.

2. Из k дополнительных вершин цвета f_2 проводятся ребра во все исходные вершины. Их количество равно kn.

3. Из $k-1$ дополнительной вершины цвета f_1 проводится $k-1$ ребро в одни и те же дополнительные вершины цвета f_2. Их количество равно $(k-1)(k-1)$.

4. Из k дополнительных вершин цвета f_2 строится полный граф K_k с количеством ребер, равным $\frac{k(k-1)}{2}$.

Доказательство. Рассмотрим, какие случаи отказа могут присутствовать в графе:

1. Удалены вершины v_{f_1} и v_{f_2}.

2. Удалены вершины v_{f_2}.

3. Удалены вершины v_{f_1} и $v_{f_1}^+$.

4. Удалены вершины v_{f_1} и $v_{f_2}^+$.

5. Удалены вершины v_{f_2} и $v_{f_2}^+$.

6. Удалены вершины v_{f_2} и $v_{f_1}^+$.

7. Удалены вершины $v_{f_1}^+$.

8. Удалены вершины $v_{f_2}^+$.

Для наглядности будем рассматривать пример построения МВ-2Р для цветного полного графа K_4, удовлетворяющего условиям теоремы. Пример подобного графа изображен на рисунке 3.2.1.

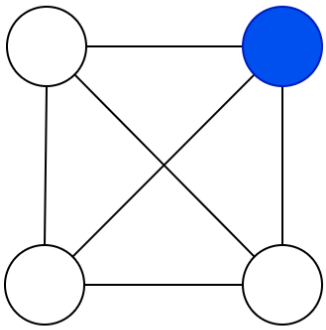

Рисунок 3.2.1 – Пример двухцветного полного графа K_4.

Последние два случая рассматривать смысла нет, поскольку в графе, полученном удалением любых k дополнительных вершин, уже содержится исходный граф. Пример данного утверждения представлен на рисунке 3.2.2.

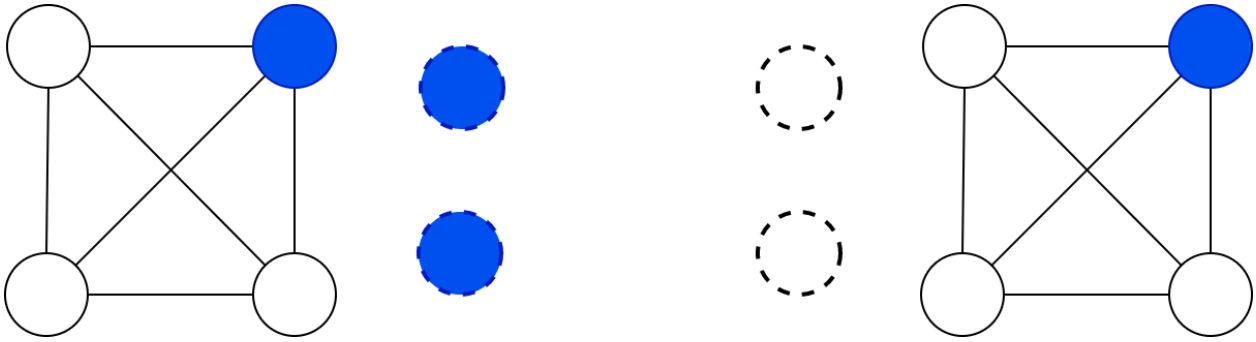

Рисунок 3.2.2 – Выполнение случаев 7 и 8 в MB-2P цветного графа K_4.

Случаи с 3 по 6 сводятся к теореме 3.1.1, поэтому количество дополнительных ребер будет равно $2n - 1$. Поскольку по условию вершинного расширения могут быть удалены любые ребра – дополнительные ребра необходимо провести из каждой вершины $v_{f_1}^+$ и $v_{f_2}^+$. Их количество равно $k(2n - 1) = k(n - 1) + kn$.

Теперь рассмотрим случай 2, когда удаляются любые k исходных вершин цвета f_2. В данном случае получившийся исходный граф становится полным графом K_{n-k}. Если рассматривать граф подобным образом, то полный граф K_n является соединением полных подграфов $K_k + K_{n-k}$. Поскольку были удалены вершины одного цвета, следовательно полный подграф K_k состоит из вершин цвета f_2. Чтобы вложить исходный граф в получившийся, необходимо восстановить соединение графа K_{n-k} с графом K_k, состоящим из вершин f_2.

Это значит, что необходимо создать граф K_k из вершин $v_{f_2}^+$ с общим количеством ребер $\frac{k(k-1)}{2}$ и соединить его с K_{n-k}, состоящим из оставшихся вершин исходного графа.

Рассмотрим данный случай на примере графа K_4 при $k = 2$. Рисунок 3.2.3 демонстрирует полный подграф графа K_4, состоящий только из исходных вершин цвета f_2.

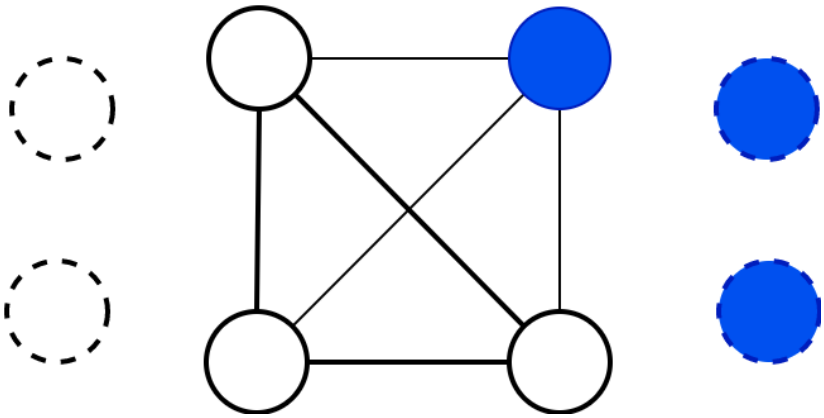

Рисунок 3.2.3 – Граф K_4, в котором выделен полный подграф K_3 цвета f_2.

Тогда рисунок 3.2.4 содержит случай удаления двух вершин v_{f_2} и граф с дополнительными ребрами в соответствии с рассуждениями выше: соединение полученного графа K_{n-2}, состоящего из вершин исходного графа, с графом K_2, состоящим из дополнительных вершин цвета f_2.

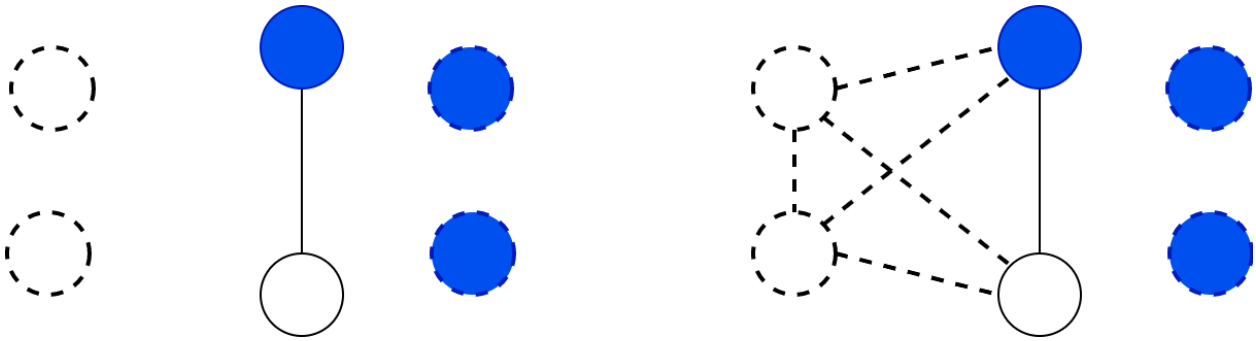

Рисунок 3.2.4 – Удаление двух случайных вершин v_{f_2} и соединение K_2 из $v_{f_2}^+$ с K_2 исходного графа.

Поскольку описанное соединение графов уже присутствует в решении для случаев 3–6, а также могут быть удалены любые вершины v_{f_2}, то общее количество дополнительных ребер для случаев 2–6 будет равно:

$$k(n-1) + kn + \frac{k(k-1)}{2}.$$

Рассмотрим оставшийся случай 1. При удалении $k-1$ вершин v_{f_2} и вершины v_{f_1} получившийся исходный граф является K_{n-k}, состоящим только из вершин цвета f_2. Таким образом, получается, что граф K_n можно рассматривать как соединение графов $K_{n-k}+K_{k-1}+K_1$, где K_{k-1} – граф из вершин цвета f_2, а K_1 – граф из вершины цвета f_1. Как и для случая 2, необходимо восстановить соединение K_{n-k} с K_{k-1} и K_1. Случаи с 3 по 6 уже имеют достаточные ребра, соединяющие v_{f_2} с $v_{f_2}^+$ и v_{f_2}с $v_{f_1}^+$ (а также включают рассмотрение графа K_{k-1}, состоящего из вершин цвета f_2). Общее число ребер будет равно:

$$\frac{(n-k)(n-k-1)}{2}+(n-k)k+\frac{(k-1)(k-2)}{2}=\frac{n^2-n-2k+2}{2}=$$

$$=\frac{n(n-1)}{2}-\frac{2k-2}{2}=\frac{n(n-1)}{2}-(k-1).$$

Несложно заметить, что данное число меньше $\frac{n(n-1)}{2}$ на $k-1$, а значит, чтобы вписать исходный полный граф в получившийся, нужно добавить $k-1$ ребер между вершинами цвета f_1 и цвета f_2. Эти ребра можно получить путем их добавления между дополнительными вершинами цвета f_1 и f_2.

Поскольку должно быть выполнено условие вершинного расширения, и максимальное количество удаленных вершин цвета f_2 в первом случае может быть только $k-1$, то для минимального вершинного k расширения достаточно провести $k-1$ дополнительных ребер из $k-1$ дополнительных вершин цвета f_2 в дополнительные вершины цвета f_1 с общим количеством $(k-1)(k-1)$. ■

Пример минимального вершинного 2-расширения для цветного графа K_4, построенного по доказанной схеме из теоремы 3.2.1, изображен на рисунке 3.2.5.

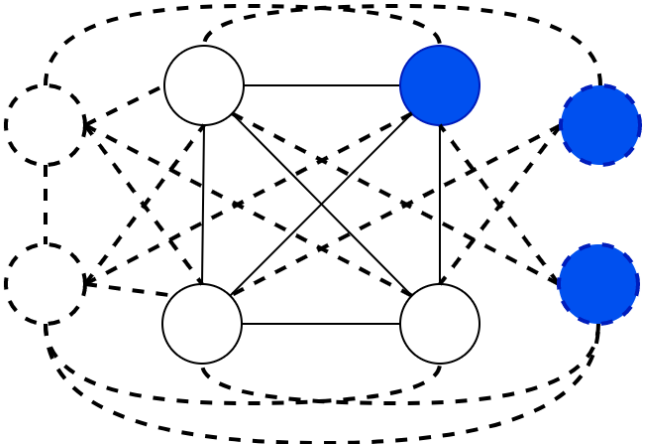

Рисунок 3.2.1 – Пример МВ-2Р для двухцветного полного графа K_4 из доказательства теоремы.

Теорема 3.2.2. Полный граф $K_n = (V, \alpha, f)$ с $F = \{f_1, f_2\}$, $|F| = 2$, $|V_{f_1}| > 1$, $|V_{f_2}| > 1$ имеет единственное с точностью до изоморфизма минимальное вершинное k-расширение с количеством дополнительных ребер, равным:

$$2nk + \frac{3k(k-1)}{2}$$

Схема построения минимального вершинного расширения:

1. Из каждой $v_{f_1}^+$ проведем ребра во все исходные вершины. Их количество равно kn.

2. Из каждой $v_{f_2}^+$ проведем ребра во все исходные вершины. Их количество равно kn.

3. Из дополнительной вершины цвета f_2 проведем ребра в $(k-1)$ дополнительную вершину цвета f_1. Далее из дополнительной вершины цвета f_1 проведем $(k-2)$ вершин в дополнительные вершины цвета f_2. Данная операция повторяется при увеличении степени k расширения, постепенно уменьшая количество дополнительных ребер. Таким образом, общее число ребер будет равно: $\sum_{j=1}^{k-1}(k-j) = \frac{k(k-1)}{2}$.

4. Из множества $v_{f_1}^+$ и множества $v_{f_2}^+$ построим два полных графа K_k с общим количеством ребер $2\frac{k(k-1)}{2}$.

Доказательство. Большинство случаев аналогичны случаям из теоремы 3.2.1, однако здесь появляется еще один дополнительный: когда удаляются k

вершин v_{f_1}. Поскольку удаление вершин v_{f_1} аналогично удалению двух v_{f_2}, то будем рассматривать этот случай как вариант, идентичный случаю 2 с оговоркой, что цвет вершин будет f_1.

Для наглядности будем рассматривать цветной граф K_4, удовлетворяющий условиям теоремы. Подобный граф изображен на рисунке 3.2.6.

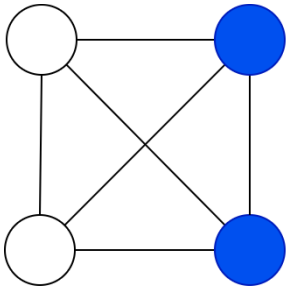

Рисунок 3.2.6 – Пример цветного полного графа K_4.

Случаи 7 и 8 так же, как и в теореме 3.2.1, не имеет смысла рассматривать.

Случаи 3–6 сводятся к теореме 3.1.2, поэтому количество дополнительных ребер будет равно $2n$. По условию вершинного расширения в графе могут отказать любые k вершин, поэтому дополнительные ребра необходимо построить из каждой дополнительной вершины. Получим $k \cdot 2n$ ребер.

Рассмотрим теперь удаление k исходных вершин одинакового цвета. Для v_{f_1} он аналогичен предыдущей теореме. Случай удаления v_{f_2} сводится к аналогичному для цвета f_1. Это означает, что необходимо построить $|F| = 2$ полных графов K_k с общим числом ребер $2\frac{k(k-1)}{2}$.

Перейдем к оставшемуся случаю, когда удаляются вершины v_{f_1} и v_{f_2}. Полученный граф содержит K_{n-k} исходных вершин различных цветов. Это означает, что нам необходимо добавить K_k, состоящий из вершин цвета f_1 и f_2. Аналогично рассуждениям в доказательстве предыдущей теоремы, чтобы вложить граф в дополнительные вершины, необходимо добавить $(k-1)$ ребер между вершинами $v_{f_1}^+$ и $v_{f_2}^+$.

При увеличении k растет число вариантов, вершины каких цветов могут быть удалены. Другими словами, растет количество вариантов, из вершин каких цветов может состоять K_k,. Например, при $k = 3$ могут быть удалены две вершины цвета f_1 и одна цвета f_2, а также может быть удалена одна вершина цвета f_1 и две – цвета f_2. Примеры такого удаления представлены на рисунке 3.2.7.

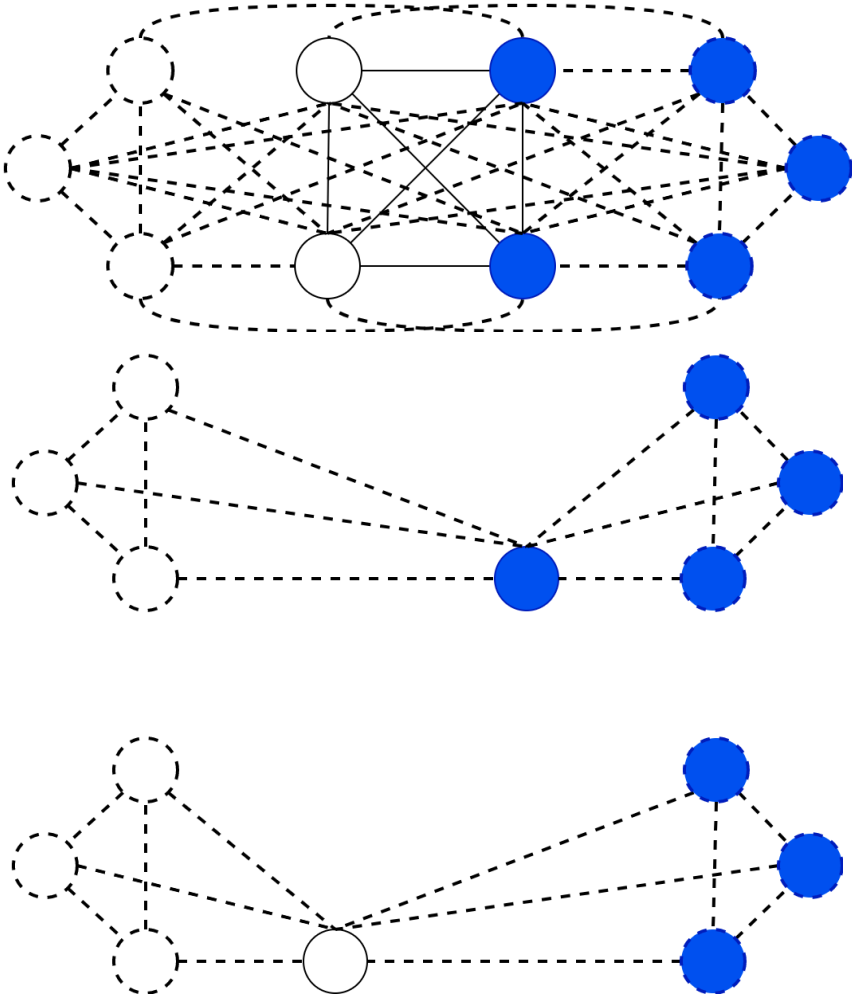

Рисунок 3.2.7 – Пример удаления из графа K_4 при $k = 3$ двух вершин цвета f_2 и одной – цвета f_1 и аналогично двух вершин цвета f_1 и одной – цвета f_1.

При этом добавление $k - 1$ ребер между дополнительными вершинами заданных цветов дают возможность вложить только один из вариантов K_k. На рисунке 3.2.8 можно заметить, что в первом случае добавление $(k - 1)$ дополнительных ребер не дает цветной вложенности исходного графа в получившийся. Вложение существует только во втором варианте удаления.

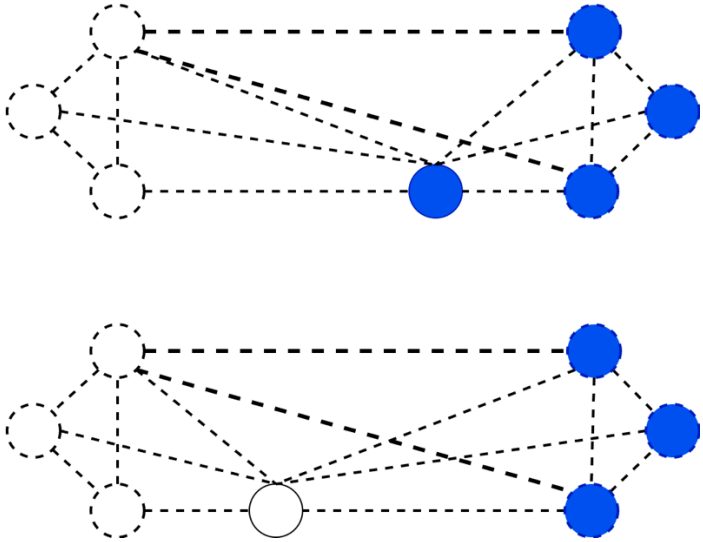

Рисунок 3.2.8 – В первом варианте условие цветного вершинного расширения для полного графа K_4 нарушается, на втором – соблюдается.

Для того, чтобы иметь возможность вложить оба варианта, а также выполнить условие минимальности, необходимо дополнительно добавить еще $k-2$ ребер между данными дополнительными вершинами разных цветов. Итоговое количество добавленных ребер будет равно $(k-1)+(k-2)$. На рисунке 3.2.9 представлено подобное добавление ребер.

При $k>3$ данная логика распространяется следующим образом: при увеличении степени расширения увеличивается количество вариантов отказов вершин, следовательно необходимо покрывать все случаи увеличением количества ребер. Так, для $k=4$ количество ребер будет равным $(k-1)+(k-2)+(k-3)$, поскольку существует три варианта удаления вершин цвета f_1 и f_2. Для $k=5$ таких вариантов четыре, и так далее. Соответственно, для покрытия всех вариантов общее число ребер будет равно $\sum_{j=1}^{k-1}(k-j)$.

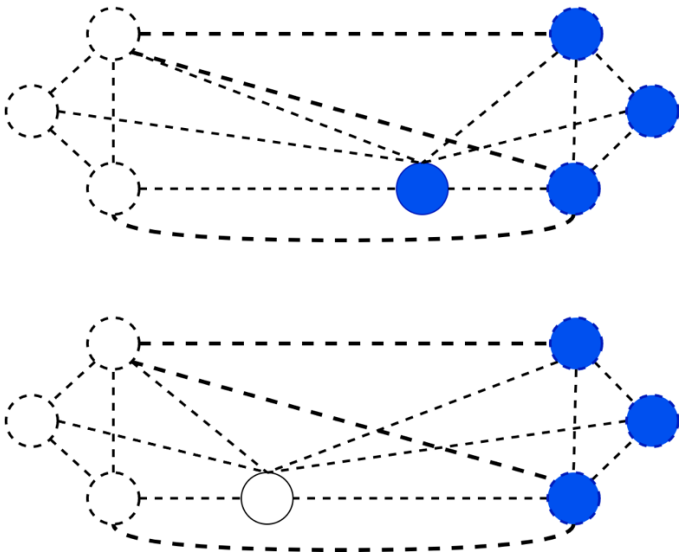

Рисунок 3.2.9 – Добавление $(k-1)+(k-2)$ дополнительных ребер в рассматриваемые графы.

Таким образом, были рассмотрены все случаи отказов вершин для полных графов заданной конфигурации. ■

На рисунке 3.2.10 содержится построенный по доказанной схеме пример МВ-2Р для цветного графа K_4 из доказательства теоремы.

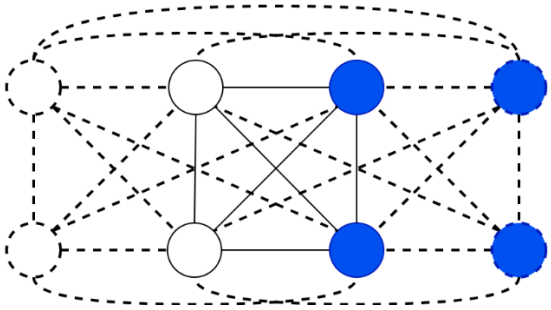

Рисунок 3.2.10 – Пример МВ-2Р цветного полного графа K_4.

3.3 Минимальные вершинные k-расширения цветных полных графов

Продолжим постепенно обобщать рассмотрение вершинных расширений для цветных полных графов. Обобщим теоремы из предыдущего параграфа на случаи при любом количестве цветов больше двух. Другими словами,

рассмотрим случай минимальных вершинных расширений для полных графов с $k \geq 1, i \geq 2$.

Несложно заметить, что для многоцветных графов выделяется четыре различных случая, связанных с уникальностью сопоставленных вершинам цветов:

1. Существует только один цвет, встречающийся в графе два или более раза, все остальные цвета единственные в графе: $|W^1| > 1, |W^*| = 1$.

2. Существует только один уникальный цвет в графе, все остальные цвета встречаются два и более раза: $|W^1| = 1, |W^*| > 1$.

3. В графе нет ни одного цвета, который бы встречался менее двух раз в заданном графе: $|W^1| = 0, |W^*| > 1$.

4. В графе нет ни одного цвета, который бы встречался два и более раза в заданном графе: $|W^1| > 1, |W^*| = 0$.

На основании выделенных случаев опишем различные схемы построения МВР для полных графов. Обобщенные части схем построения каждого рассматриваемого расширения уже были доказаны ранее и, по сути, являют собой комбинации схем построения для частных случаев.

Теорема 3.3.1. Полный граф $K_n = (V, \alpha, f)$ с $|F| \geq 2, |W^1| > 1, |W^*| = 1$ имеет единственное с точностью до изоморфизма минимальное вершинное k-расширение с количеством дополнительных ребер, равным:

$$|W^1|(n - |W^1|)k + nk + |W^1|(k-1)^2 + \frac{k(k-1)}{2} + k$$

Схема построения данного минимального вершинного расширения будет следующей:

1. Из k вершин $v^+_{W^1}$ (для каждого цвета из $F(W^1)$) проводится $(n-2)$ ребер в вершины исходного графа v_{W^*} – во все вершины кроме вершин цвета из $F(W^1)$. Количество ребер: $|W^1|(n - |W^1|)k$.

2. Из k дополнительных вершин $v^+_{W^*}$ проводятся ребра во все вершины исходного графа. Количество ребер: nk.

3. Из $k-1$ дополнительной вершины $v^{+}_{W^1}$, для каждого цвета из $F(W^1)$ проводится $k-1$ ребро в одни и те же дополнительные вершины цвета из $F(W^*)$. Количество ребер: $|W^1|(k-1)(k-1)$.

4. Каждая пара дополнительных вершин разного цвета из $F(W^1)$, соединяется ребром, итого k дополнительных ребер.

5. Строится полный граф из дополнительных вершин цвета из $F(W^*)$ с общим количеством ребер $\frac{k(k-1)}{2}$.

Доказательство. Следует из теоремы 3.2.1. Четвертый пункт схемы обосновывается тем фактом, что при удалении любого набора, состоящего из вершин из W^1, теряется связь между вершинами данного цвета, поэтому его нужно сконструировать из дополнительных вершин. ■

Рисунок 3.3.1 содержит пример трехцветного полного графа и его минимальное вершинное 2-расширение, полученное по представленной схеме.

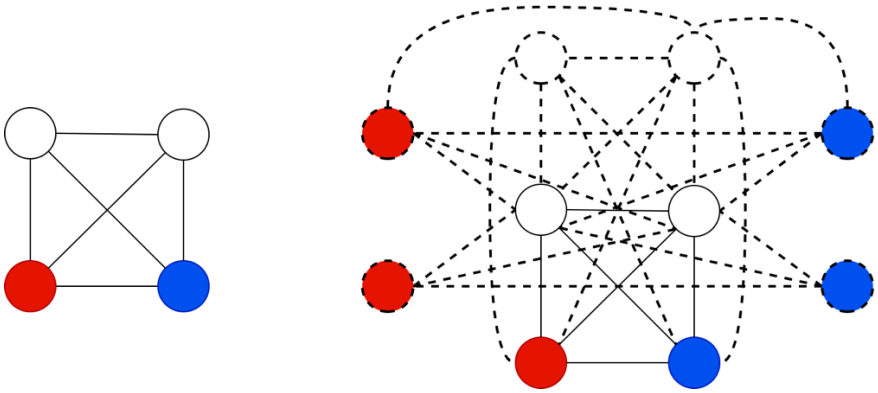

Рисунок 3.3.1 – Пример трехцветного полного графа K_4 и его МВ-2Р.

Теорема 3.3.2. Полный граф $K_n = (V, \alpha, f)$ с $|F| \geq 3$, $|W^1| = 1$, $|W^*| > 1$ имеет единственное с точностью до изоморфизма минимальное вершинное k-расширение с количеством дополнительных ребер, равным:

$$(n-1)k + |W^*|nk + |W^*|(k-1)^2 + (|W^*|+1)\frac{k(k-1)}{2}$$

Схема построения данного минимального вершинного расширения:

1. Из k дополнительных вершин $v^{+}_{W^1}$ проводятся ребра во все вершины v_{W^*}. Количество ребер: $(n-1)k$.

2. Из k вершин $v^+_{W^*}$ каждого цвета из $F(W^*)$ проводятся ребра во все вершины исходного графа. Количество ребер: $|W^*|nk$.

3. Из $k-1$ дополнительной вершины $v^+_{W^1}$, для каждого цвета из $F(W^1)$ проводится $k-1$ ребро в одни и те же дополнительные вершины цвета из $F(W^*)$. Количество ребер: $|W^*|(k-1)(k-1)$.

4. Из дополнительной вершины $v^+_{W^*}$ цвета $f_1 \in F(W^*)$ проведем ребра в $(k-1)$ дополнительную вершину цвета $f_2 \in F(W^*)$. Далее из дополнительной вершины цвета f_2 проведем $(k-2)$ вершин в дополнительные вершины цвета f_1. Данная операция повторяется при увеличении степени k расширения, постепенно уменьшая количество дополнительных ребер. Таким образом, общее число ребер будет равно: $\sum_{j=1}^{k-1}(k-j) = \frac{k(k-1)}{2}$.

5. Для каждого цвета из $F(W^*)$ построим полный граф из $v^+_{W^*}$ с числом ребер $|W^*|\frac{k(k-1)}{2}$.

Доказательство. Доказательство данной схемы можно разбить на две – схема построения расширения для графа с $|W^1| = 1, |W^*| > 1$ и схема для $|W^1| \geq 1, |W^*| = 1$. Пункты 1–3 следуют из теоремы 3.2.1. Последние два пункта следуют из теоремы 3.2.2. ■

Пример трехцветного графа и его минимального вершинного 1-расширения для теоремы 3.3.2 представлены на рисунке 3.3.2.

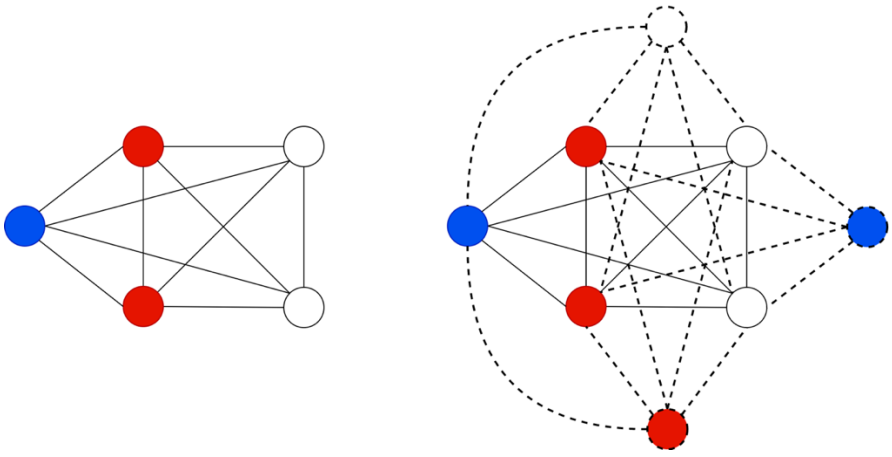

Рисунок 3.3.2 – Пример трехцветного графа K_4 и его МВ-1Р, построенного по схеме теоремы 3.3.2.

Следствие 3.3.1. Схема построения минимального вершинного расширения из теоремы 3.3.2 справедлива для полных графов $K_n = (V, \alpha, f)$ с $|W^1| \geq 1$, $|W^*| > 1$ с общим количеством дополнительных ребер, равным:

$$(n-1)k + |W^*|nk + |W^*| \cdot |W^1|(k-1)^2 + (|W^*|+1)\frac{k(k-1)}{2}$$

Доказательство. Поскольку в пункте 3 схемы построения из теоремы 3.3.2 оговаривается, что из каждой дополнительной вершины $v^+_{W^1}$ проводятся ребра в дополнительные вершины цвета $F(W^*)$, то несложно заметить, что при увеличении мощности множества W^1 каждая дополнительная вершина $v^+_{W^1}$ должна иметь $(k-1)(k-1)$ ребер в дополнительные вершины цвета $F(W^*)$. Таким образом, общее число дополнительных ребер для третьего пункта схемы построения становится равным $|W^*| \cdot |W^1|(k-1)(k-1)$. ■

Теорема 3.3.3. Полный граф $K_n = (V, \alpha, f)$ с $|F| \geq 2$, $|W^1| = 0$, $|W^*| > 1$ имеет единственное с точностью до изоморфизма минимальное вершинное k-расширение с количеством дополнительных ребер, равным:

$$|W^*|nk + \frac{|W^*|(|W^*|+1)}{2} \cdot \frac{k(k-1)}{2}$$

Схема построения минимального вершинного расширения такого графа будет следующей:

1. Для каждого цвета из $F(W^*)$ из k дополнительных вершин $v^+_{W^*}$ проводятся ребра во все вершины исходного графа. Количество ребер: $|W^*|nk$.

2. Для каждой пары цветов $f_1, f_2 \in F(W^*)$ из дополнительной вершины цвета f_1 проводятся $k-1$ ребер в вершины цвета f_2, а из дополнительной вершины цвета f_2 ребра в $k-2$ дополнительные вершины цвета f_1 и так далее при увеличении степени расширения. Несложно заметить, что все рассматриваемые пары образуют полный $|W^*|$-вершинный граф. Таким образом, в общем случае количество дополнительных ребер будет равным $\frac{|W^*|(|W^*|-1)}{2}\left(\sum_{j=1}^{k-1}(k-j)\right) = \frac{|W^*|(|W^*|-1)}{2} \cdot \frac{k(k-1)}{2}$.

3. Для каждого цвета из $F(W^*)$ строится полный граф из $v^+_{W^*}$ с числом ребер $|W^*|\frac{k(k-1)}{2}$.

Доказательство. Следует из теорем 3.2.2, 3.3.1 и 3.3.2. ■

Пример графа, который удовлетворяет условиям теоремы 3.3.3, а также его МВ-1Р, построенной по схеме, изображены на рисунке 3.3.3.

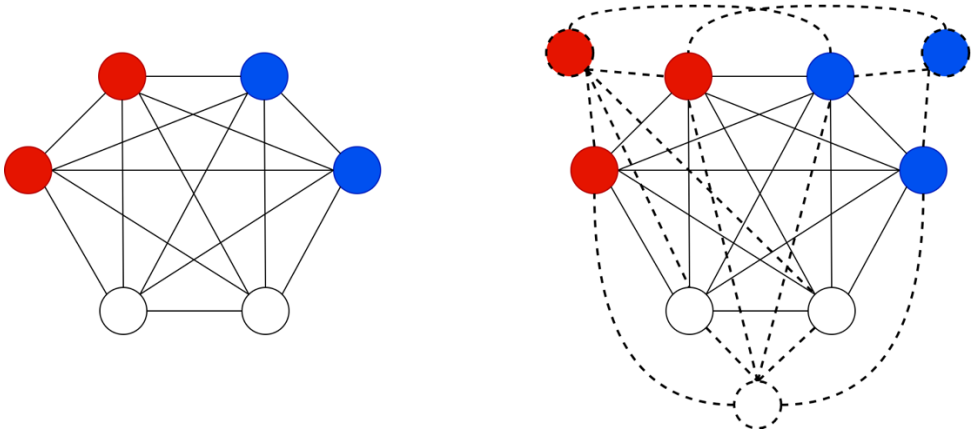

Рисунок 3.3.3 – Пример трехцветного графа K_6 и его МВ-1Р.

Теорема 3.3.4. Полный граф $K_n = (V, \alpha, f)$ с $|F| \geq 2$, $|W^1| > 1$, $|W^*| = 0$ имеет единственное с точностью до изоморфизма минимальное вершинное k-расширение с количеством дополнительных ребер, равным:

$$k\frac{|W^1|(|W^1|-1)}{2}$$

Схема построения заключается в том, чтобы построить k полных графов из вершин $v^+_{W^1}$.

Доказательство. При удалении любой вершины любого цвета граф K_n превращается в K_{n-1}, поэтому необходимо соединить K_{n-1} с K_1, построенный из дополнительной вершины удаленного цвета. Поскольку может быть удалена вершина любого цвета и для каждого цвета необходимо соединять K_1 удаленного цвета с K_{n-1}, то нужно провести $n-1$ ребер из каждой дополнительной вершины. Очевидно, что условие минимальности расширения достигается тогда, когда каждая дополнительная вершина соединена с дополнительными вершинами остальных цветов. Всего получится $\frac{n(n-1)}{2}$ ребер. С учетом степени

расширения k, необходимо построить k полных графов из k дополнительных вершин. ■

Пример подобного полного графа и его минимального вершинного 1-расширения изображены на рисунке 3.3.4.

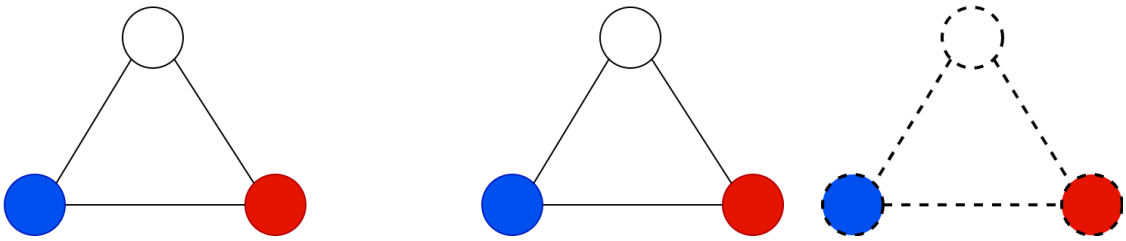

Рисунок 3.3.4 – Пример трехцветного графа K_3 и его МВ-1Р.

Рассматривая представленные теоремы, можно обнаружить некоторую закономерность в построении схемы минимального вершинного k-расширения для различных конфигураций графов. В действительности, все описанные выше схемы можно свести к единой и обобщить все теоремы выше к общей теореме для цветных полных графов любой конфигурации.

3.4 Общая схема построения минимальных вершинных k-расширений цветных полных графов

Следующая теорема является следствием закономерного развития идеи построения схем минимального вершинного k-расширения для цветных полных графов, рассмотренных в теоремах 3.1.1, 3.1.2, 3.2.1, 3.2.2 и 3.3.1–3.3.4 главы 3.

Теорема 3.4.1. Полный граф $K_n = (V, \alpha, f)$ с $|F| > 1$ имеет единственное с точностью до изоморфизма минимальное вершинное k-расширение с количеством дополнительных ребер, равным:

$$|W^1|(n - |W^1|)k + |W^*|nk + |W^*| \cdot |W^1|(k-1)^2 +$$

$$+\frac{|W^*|(|W^*|+1)}{2}\cdot\frac{k(k-1)}{2}+k\frac{|W^*|(|W^*|-1)}{2}$$

Схема построения данного минимального вершинного расширения заключается в следующем:

1. Для каждого цвета из $F(W^1)$ проводятся ребра из k дополнительных вершин $v_{W^1}^+$ во все вершины v_{W^*}. Количество ребер: $|W^1|(n-|W^1|)k$.

2. Для каждого цвета из $F(W^*)$ из k дополнительных вершин $v_{W^*}^+$ проводятся ребра во все вершины исходного графа. Количество ребер: $|W^*|nk$.

3. Из $k-1$ дополнительной вершины $v_{W^*}^+$, для каждого цвета из $F(W^*)$ проводится $k-1$ ребро в одни и те же дополнительные вершины цвета из $F(W^1)$. Количество ребер: $|W^*|\cdot|W^1|(k-1)(k-1)$.

4. Для каждой пары цветов $f_1, f_2 \in F(W^*)$ из дополнительной вершины цвета f_1 проводятся $k-1$ ребер в вершины цвета f_2, а из дополнительной вершины цвета f_2 ребра в $k-2$ дополнительные вершины цвета f_1 и так далее при увеличении количества вариантов удаления исходных вершин различного цвета. Общие число ребер будет равно $\frac{|W^*|(|W^*|-1)}{2}\left(\sum_{j=1}^{k-1}(k-j)\right)=\frac{|W^*|(|W^*|-1)}{2}\cdot\frac{k(k-1)}{2}$.

5. Для каждого цвета из $F(W^*)$ строится полный граф из $v_{W^*}^+$ с общим числом ребер $|W^*|\frac{k(k-1)}{2}$.

6. Каждая пара дополнительных вершин разного цвета из $F(W^*)$ соединяется ребром. Общее количество ребер: $k\frac{|W^*|(|W^*|-1)}{2}$.

Доказательство. Следует из теорем 3.3.1, 3.3.3, 3.3.4, а также замечания 3.3.2. ■

Исследуем утверждение, что теорема 3.4.1 справедлива и для непомеченных полных графов. Приведем теорему из работы М.Б. Абросимова [1].

Теорема 3.4.2. Минимальное вершинное k-расширение, причем единственное с точностью до изоморфизма, полного n-вершинного графа K_n есть полный $(n+k)$-вершинный граф K_{n+k}.

Утверждение. Схема построения минимального вершинного k-расширения из теоремы 3.4.1 с заданным по формуле количеством дополнительных ребер справедлива и для полных графов с $|F| = 1$.

Доказательство. Представим непомеченный полный граф как цветной граф с одним цветом $K_n = (V, \alpha, f)$ с $|F| = 1$, $|W^1| = 0$, $|W^*| = 1$. Посчитаем количество дополнительных ребер в расширении K_{n+k}:

$$\frac{(n+k)(n+k-1) - n(n-1)}{2} = \frac{n^2 + kn - n + kn + k^2 - k - n^2 + n}{2} =$$

$$= \frac{2kn + k^2 - k}{2} = kn + \frac{k(k-1)}{2}.$$

Теперь посчитаем количество дополнительных ребер по теореме 3.4.1.

$$0(n-0)k + nk + 0 + \frac{1 \cdot 2}{2} \cdot \frac{k(k-1)}{2} + k\frac{0 \cdot (-1)}{2} =$$

$$= kn + \frac{k(k-1)}{2}.$$

Таким образом, теорема 3.4.1 справедлива и для непомеченных полных графов. ■

Глава 4. Минимальные расширения цветных звездных графов

В данной главе приводятся схемы построения минимальных расширений для звездных графов, и показано, что покрываются все возможные виды звездных графов с введенной на них функцией раскраски.

В главе 4 будут использоваться введенные в главе 3 обозначения. Кроме этого, вводится еще одно обозначение: множество $V_c = \{v | v \in V, f(v) = f(c)\}$ будет обозначать набор вершин, цвет которых совпадает с цветом центральной вершины. Также следует отметить, что звездный граф может быть определен как соединение одновершинного полного графа со вполне несвязным: $K_1 + O_n$, где K_1 – полный граф с одной вершиной, а O_n – вполне несвязный граф с n вершинами.

4.1 Минимальные реберные расширения цветных звезд

Для простых непомеченных звезд М.Б. Абросимов доказал следующую теорему [1]:

Теорема 4.1.1. При $k \leq n/2$ граф $K_{1+2n} + O_{n-2k}$ является единственным с точностью до изоморфизма минимальным реберным k-расширением звездного графа $K_1 + O_n$. При $k > n/2$ звезда $K_1 + O_n$ не имеет минимальных реберных k-расширений.

Обобщим теорему 4.1.1 на случай произвольных цветных звезд. Оказывается, что результат переносится почти без существенных изменений.

Рассмотрим различные конфигурации неизоморфных цветных звезд и возможность построения MP-kP для них. Достаточно очевидно, что цветные звездные графы с определенной мощностью V_c не имеют реберных расширений.

Теорема 4.1.2. Любой цветной звездный граф $S = (V, \alpha, f)$, у которого $|V_c| < 1 + 2k$, не имеет минимального реберного k-расширения.

Доказательство. Очевидно, при удалении k ребер графа не найдется достаточного количества дополнительных ребер, способных заменить ребра исходного графа. Соответственно, невозможно вписать исходный граф в граф, получающийся удалением k ребер из S^*. ■

Рассмотрим поиск реберного расширения на примере MP-2P. Для MP-2P $k = 2$, а $2k + 1 = 5$. Допустим цветной звездный граф содержит $|V_c| = 4$ вершин цвета $f(c)$ – на одну меньше, чем $2k + 1$. Пример графа изображен на рисунке 4.1.1. В исходном графе суммарное количество ребер между всеми вершинами из V_c равно трем.

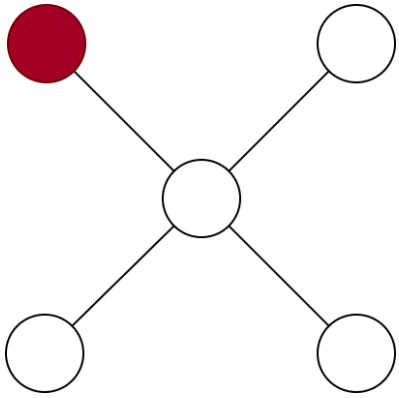

Рисунок 4.1.1 – Пример цветной звезды с $|V_c| = 4$.

Максимального суммарного количества ребер между вершинами из V_c можно добиться, построив полный подграф из вершин цвета $f(c)$. При этом количество ребер будет равно:

$$\frac{|V_c|(|V_c| - 1)}{2} = \frac{4(4 - 1)}{2} = 6.$$

Пример графа с полным подграфом из вершин множества V_c представлен на рисунке 4.1.2. В данном графе каждая вершина цвета $f(c)$ соединена с тремя вершинами того же цвета. Тогда при удалении $k = 2$ ребер таким образом, чтобы степень каждой вершины цвета $f(c)$ данного графа стала равна двум, количество смежных вершин цвета $f(c)$ каждой вершине того же цвета становится равной двум.

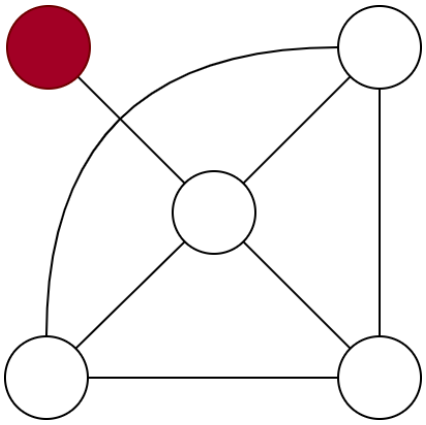

Рисунок 4.1.2 – Пример полного подграфа из вершин цвета $f(c)$.

Пример удаления ребер подобным образом изображен на рисунке 4.1.3.

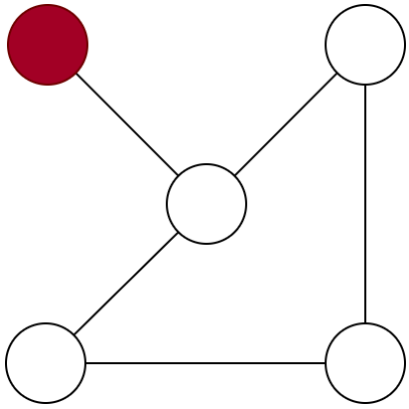

Рисунок 4.1.3 – Пример удаления 2 ребер из графа с полным подграфом цвета $f(c)$.

Это означает, что исходный цветной граф невозможно вложить в получившийся после удаления ребер. Поскольку мы рассмотрели полный подграф из вершин цвета $f(c)$, то и для графов с количеством дополнительных ребер меньше, чем у полного, рассуждения справедливы.

Теперь допустим, что количество вершин цвета центральной вершины равно пяти: $|V_c| = 5$. Это означает, что каждая вершина цвета $f(c)$ соединена

смежна четырем вершинам того же цвета. Суммарное количество ребер полного подграфа из вершин множества V_c равно $\frac{5(5-1)}{2} = 10$. Вектор степеней данного полного подграфа будет равным $(4, 4, 4, 4, 4)$. Тогда при удалении любых двух ребер вектор степеней этого подграфа в худшем случае может стать равным $(4, 3, 3, 3, 3)$. Данный вывод означает, что найдется хотя бы одна вершина из V_c, соединенная со всеми другими вершинами из V_c.

Исходя из теоремы 4.1.2, для графов с $|V_c| < 3$ не существует МР-1Р, для $|V_c| < 5$ не существует МР-2Р, для $|V_c| < 7$ не существует МР-3Р и так далее.

Теорема 4.1.3. Рассмотрим звезду $S = (V, \alpha, f)$, у которой множество $|V_c| \geq 2k + 1$, а общее число вершин $|V| = n + 1$. Минимальным реберным k-расширением графа S, причем единственным с точностью до изоморфизма, будет являться цветной граф с общим количеством ребер равным:

$$(n - k)(2k + 1)$$

Схема построения данного МР-kР заключается в построении $2k$ дополнительных центров из вершин $v \in V_c, v \neq c$.

Доказательство. Создание дополнительного центра является достаточно очевидным шагом при построении расширения, поскольку при удалении любого ребра из исходного множества α вершина c перестает быть центром.

Данную операцию построения нового центра из вершины цвета $f(c)$, не являющуюся центром, необходимо произвести $2k$ раз.

Допустим, в исходный граф S было построено описанным образом k дополнительных центров. Выбираем $k - 1$ пару дополнительных центров таким образом, чтобы никаких две пары не содержали одинаковой вершины. Тогда при удалении ребра между элементами каждой пары вершины пар перестают быть центрами, и в графе остается только один центр – вершина c. Удаляя оставшееся ребро между вершиной c и любой другой вершиной, мы нарушаем условие вложенности с учетом сохранности цветов.

Повторим данные рассуждения до построения $2k-1$ дополнительных центров. При построении $k-1$ пары будут задействованы $2(k-1)$ дополнительные вершины. Поскольку по допущению мы строим $2k-1$ дополнительных центров, то останется ровно один дополнительный центр. Тогда при удалении $k-1$ ребра между элементами $k-1$ пары дополнительных центров, а также при удалении одного ребра между исходным центром и оставшимся дополнительным центром, условие цветной вложенности нарушается.

Тогда при построении $2k$ дополнительных центров, после разбиения $2(k-1)$ дополнительных центров на пары в графе останется в точности две новые центральные вершины, не попавшие в $k-1$ пару. Поскольку после удаления $k-1$ ребер между элементами каждой пары останется возможным удалить только одно ребро, значит останется как минимум одна вершина, оставшаяся дополнительным центром. Из этого следует, что получившийся граф допускает цветное вложение исходного. ■

Рассмотрим ход рассуждений доказательства теоремы 4.1.3 на графе $S = K_{1,5}$. Пример подобного графа, удовлетворяющего условиям теоремы, представлен на рисунке 4.1.4.

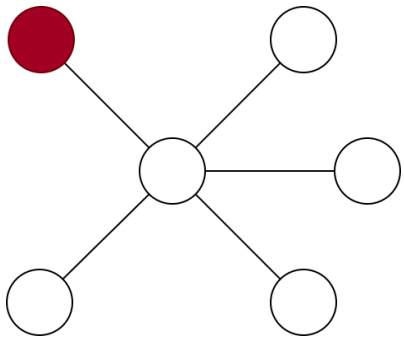

Рисунок 4.1.4 – Пример цветной звезды для теоремы 4.1.3.

Допустим, необходимо найти MP-2P для звезды из примера, то есть $k = 2, |V| = 6, |\alpha| = 4$, при этом $|V_c| = 5$. Строим дополнительный центр из $v \in V_c, v \neq c$. В получившемся графе теперь два центра: c и v. Однако несложно заметить, что в данном графе есть ребро (v, c), удаление которого приведет к графу, в который невозможно вписать исходный граф. Чтобы удовлетворить условиям реберного расширения, необходимо добавить дополнительное

ребро, дублирующее (v, c). Единственным возможным вариантом добавления данного ребра с учетом соблюдения условий реберного расширения является создание еще одного центра из еще одной вершины $u \in V_c, u \neq c, u \neq v$. Таким образом, при удалении ребра (v, c) его заменяет ребро (u, c) и наоборот. Пример для $k = 2$, $|V_c| = 5$ представлен на рисунке 4.1.5.

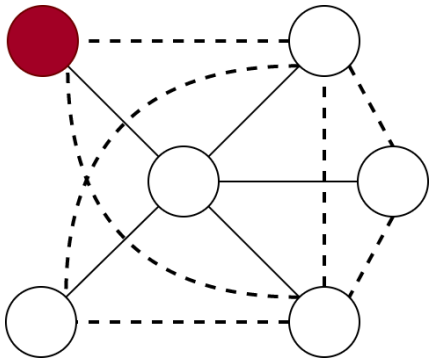

Рисунок 4.1.5 – Пример построенного графа с $k = 2$ дополнительными центрами.

Создадим только k дополнительных центров путем добавления дополнительных ребер. Далее разобьем центральные вершины в графе на $k - 1$ пару, как описано в доказательстве. Тогда, при удалении $k - 1$ ребер между парами центральных вершин, а также ребра, инцидентного исходному центру, нарушается условие цветной вложенности. Пример подобного удаления $k - 1$ ребер между k дополнительными центрами и одним ребром между вершиной c и любой другой вершиной представлен на рисунке 4.1.6.

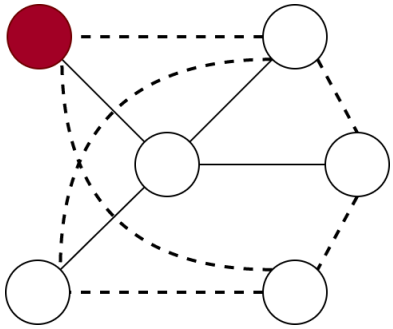

Рисунок 4.1.6 – Пример удаления двух ребер из полученного графа описанным способом.

Повторим рассуждения до $2k - 1$. Как и для k дополнительных центров, при удалении k ребер из построенного графа условие цветной вложенности нарушается. Пример подобного удаления представлен на рисунке 4.1.7.

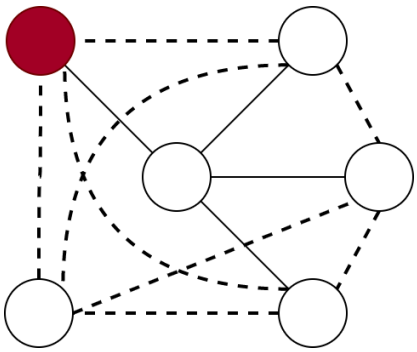

Рисунок 4.1.7 – Пример удаления k ребер из графа с $2k-1$ дополнительным центром.

При построении же $2k$ дополнительных центров, останется как минимум одна вершина, смежная всем остальным – это и будет центр. Таким образом, итоговое MP-2P будет содержать $2k$ дополнительных центров. Пример MP-2P изображен на рисунке 4.1.8.

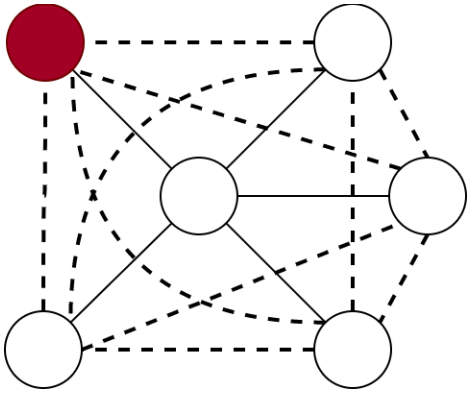

Рисунок 4.1.8 – Минимальное реберное 2-расширение для цветной звезды.

Количество ребер в MP-kP легко подсчитать. Пусть звезда S имеет количество вершин равное $n+1$ (то есть одну центральную вершину и n периферийных). Тогда в ее MP-kP при условии выполнения теоремы 4.1.3 будет $2k+1$ вершин со степенью n и остальные $n-2k$ вершин будут иметь степень $2k+1$. Тогда общее число ребер будет равно $(n-k)(2k+1)$.

Замечание 4.1.1. Теоремы 4.1.2 и 4.1.3 полностью решают вопрос о минимальных реберных k-расширениях для любых цветных звезд.

4.2 Минимальные вершинные расширения цветных звезд

Перейдем теперь к рассмотрению схем построения для минимальных вершинных k-расширений цветных звезд.

Схемы построения МВ-kР для цветных звездных графов можно разбить на два случая: когда в звезде S мощность множества $|V_c| = 1$ и когда она равна $|V_c| > 1$. Первый случай разбивается на несколько подзадач:

– $|V_c| = 1$, $|W^1| > 1$, $|W^*| = 0$ – ни один цвет не встречается в графе дважды.

– $|V_c| = 1$, $|W^1| = 1$, $|W^*| \geq 1$ – все цвета, кроме цвета $f(c)$, встречаются в графе как минимум дважды.

– $|V_c| = 1$, $|W^1| > 1, |W^*| \geq 1$ – граф содержит как вершины с уникальными цветами, так и вершины, цвет которых встречается в графе более одного раза.

Отдельным вариантом является цветная звезда с $|V_c| > 1$. Его мы рассмотрим после всех возможных конфигураций звезд с $|V_c| = 1$, поскольку он объединяет в себе различные результаты для более простых конфигураций и является обобщением для схем построения минимальных вершинных расширений цветных звезд.

Начнем рассматривать выделенные случаи, постепенно переходя к обобщению получающихся схем.

Теорема 4.2.1. Звезда $S = (V, \alpha, f)$, $|V| = n$, у которой только центральная вершина имеет цвет $f(c)$: $|V_c| = 1$, а также $|W^1| > 1$, $|W^*| = 0$, имеет единственное с точностью до изоморфизма минимальное вершинное k-расширение, содержащее

$$(|W^1| - 1)k$$

дополнительных ребер.

Схема построения МВ-kР заключается в том, что из k дополнительных вершин строится k исходных графов.

Доказательство. Рассмотрим случай МВ-1Р, то есть $k = 1$. Для наглядности приведем пример звезды, удовлетворяющей условиям теоремы, с дополнительными вершинами. Подобная звезда представлена на рисунке 4.2.1.

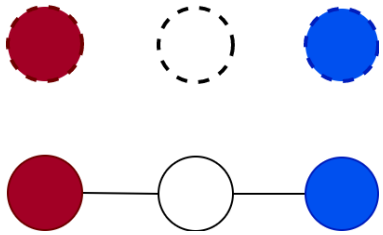

Рисунок 4.2.1 – Трехцветная звезда с дополнительными вершинами.

При удалении центральной вершины граф превращается в $(n-1)$ изолированную вершину различных цветов. Тогда достаточно провести $(n-1)$ ребро из дополнительной вершины цвета $f(c)$ к вершинам отличного от $f(c)$ цвета, чтобы иметь возможность вложить в него исходный граф. Пример удаления центральной вершины c и добавления $(n-1)$ ребра из дополнительной вершины цвета $f(c)$ представлен на рисунке 4.2.2.

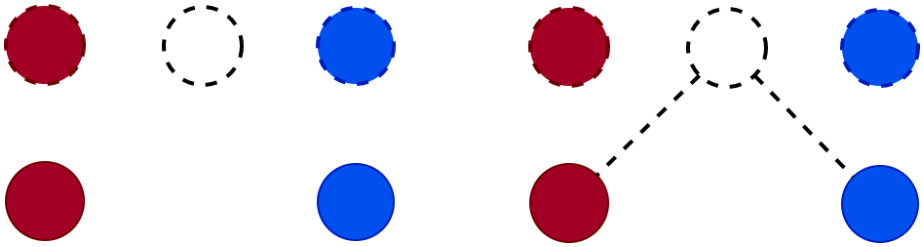

Рисунок 4.2.2 – Пример графа с удаленной вершиной c и $(n-1)$ ребром из вершины цвета $f(c)$.

При удалении какой-либо вершины, отличной от c, для вложения исходного графа достаточно провести ребро из дополнительной вершины того же цвета в вершину цвета $f(c)$, соединенную с вершинами всех остальных цветов. Описанные действия представлены на примере звезды и изображены на рисунке 4.2.3.

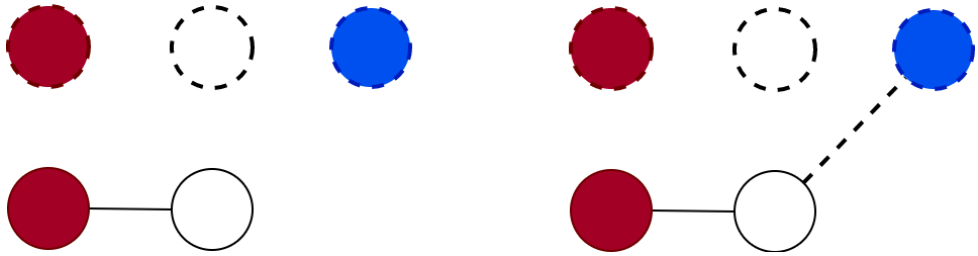

Рисунок 4.2.3 – Граф с удаленной нецентральной вершиной и граф с дополнительным ребром.

Исходя из полученных данных, общее количество ребер получается $(n-1)$ для дополнительной вершины цвета $f(c)$ и $(n-1)$ ребер для вершин отличного от цвета $f(c)$ к вершинам цвета $f(c)$. Очевидно, что ребра из вершин цвета $f(c)$ и ребра в вершины цвета $f(c)$ – это одни и те же ребра. Таким образом, общее необходимое количество дополнительных ребер будет равно $n-1$. А, поскольку $|W^1| = |F| = |V|$, то получаем

$$|W^1| - 1.$$

Аналогичные рассуждения приводятся для $k = 2,3, \ldots$

Таким образом, общее количество дополнительных ребер для МВ-kР будет равно $(|W^1| - 1)k$. ■

Пример удовлетворяющего условиям теоремы 4.2.1 звездного графа и МВ-2Р, построенного по представленной схеме, изображены на рисунке 4.2.4.

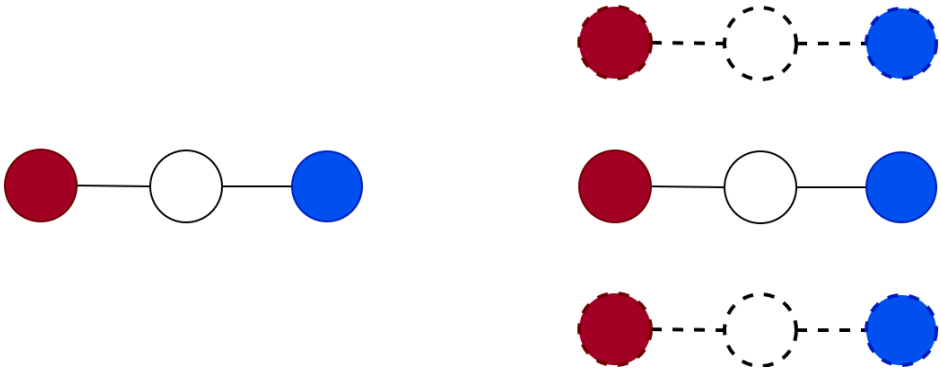

Рисунок 4.2.4 – Трехцветная звезда и ее МР-2Р, построенное по схеме теоремы 4.2.1.

Теорема 4.2.2. Звезда $S = (V, \alpha, f)$, где $|V_c| = 1$, $|W^1| = 1$, $W^* = \{W_0^*\}$, $|W^*| = 1$ имеет, причем единственное с точностью до изоморфизма, минимальное вершинное k-расширение, содержащее

$$k(|W_0^*| + k)$$

дополнительных ребер.

Схема построения заключается в том, что из всех дополнительных вершин $f(c)$ цвета проводятся ребра во все исходные и дополнительные вершины цвета $F(W_0^*)$.

Доказательство. Пусть цвет вершины из W_0^* обозначается как f_1.

Рассмотрим случай, когда $k = 1$. При удалении вершины c теряются $|W_0^*|$ ребер. Соответственно, для удовлетворения условиям вершинного расширения необходимо добавить $|W_0^*|$ ребро между вершиной $f(c)$ цвета и вершинами цвета f_1. Иллюстрация рассуждений для графа $K_{1,3}$ представлена на рисунке 4.2.5.

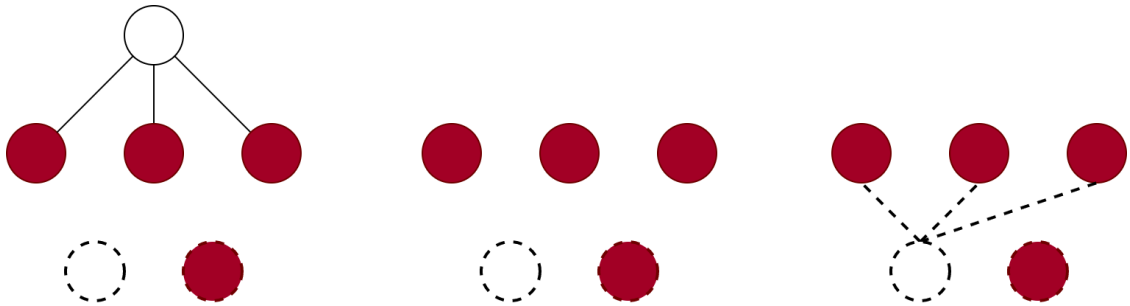

Рисунок 4.2.5 – Пример удаления вершины c из цветной звезды и добавление дополнительных ребер при $k = 1$.

При удалении вершины с цветом f_1, количество ребер между вершинами данного цвета и вершинами цвета $f(c)$ уменьшается на 1. Поэтому необходимо добавить 1 дополнительное ребро. Наглядная иллюстрация рассуждений для заданной на рисунке 4.2.5 звезды изображена на рисунке 4.2.6.

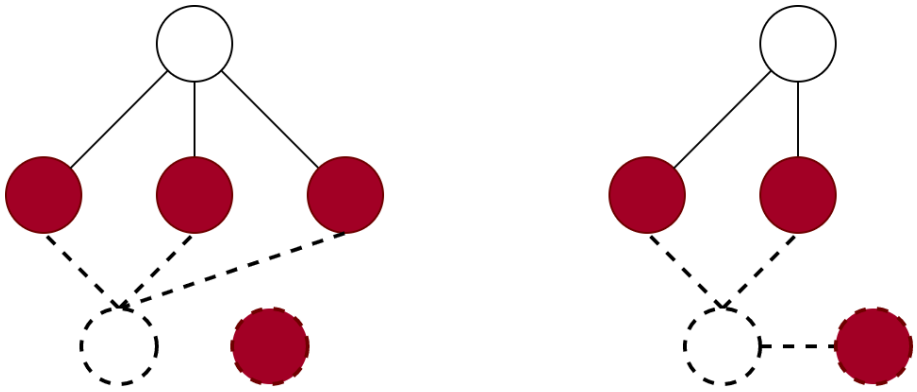

Рисунок 4.2.6 – Пример удаления вершины цвета f_1 и добавления дополнительного ребра.

Теперь рассмотрим случаи $k > 1$. Здесь может произойти несколько случаев:

1) удалено k вершин цвета $f(c)$;

2) удалено k вершин цвета f_1;

3) удалено k вершин, одна часть которых имеет цвет $f(c)$, а другая – цвет f_1 соответственно.

В первом случае необходимо создать дополнительный центр из вершины цвета $f(c)$, при этом количество дополнительных ребер будет равно $|V_{f_1}| = |W_0^*|$. Поскольку условие вершинного расширения предполагает, что могут быть удалены любые вершины, то количество дополнительных центров должно быть равно k, то есть из каждой дополнительной вершины цвета $f(c)$ необходимо построить дополнительный центр с общим количество ребер $k|W_0^*|$.

Во втором случае необходимо провести k ребер в k дополнительных вершины цвета f_1 из центра, содержащего $|W_0^*| - k$ ребер в вершины цвета f_1.

Третий вариант будет удовлетворен в случае, если решение первого варианта дополнить решением второго варианта, то есть из k дополнительных центров цвета $f(c)$ провести по k ребер в k дополнительные вершины цвета f_1. При этом количество дополнительных ребер из вершин цвета $f(c)$ в вершины f_1 будет равно $k \cdot k$.

Таким образом, общее количество дополнительных ребер будет равно $k|W_0^*| + k \cdot k = k(|W_0^*| + k)$. ■

Пример звезды, удовлетворяющей условиям, и ее минимального вершинного 2-расширения, построенного по описанной схеме, представлены на рисунке 4.2.7.

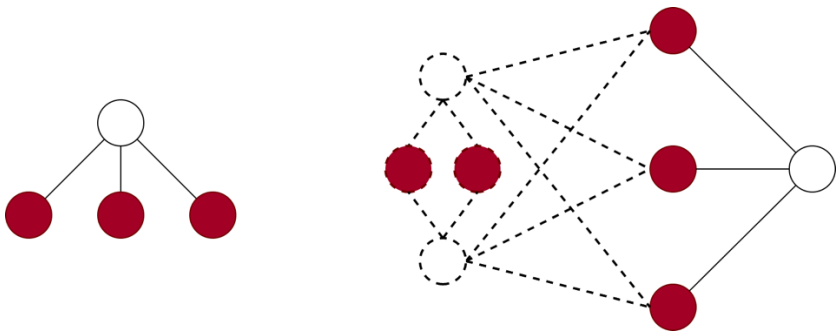

Рисунок 4.2.7 – Пример 4-вершинного двухцветного звездного графа и его МВ-2Р.

Следствие 4.2.1. Звезда $S = (V, \alpha, f)$, у которой $|V_c| = 1, |W^1| = 1, W^* = \{W_1^*, W_2^*, W_3^*, \dots, W_r^*\}$, $|W^*| \geq 1$, имеет единственное с точностью до изоморфизма минимальное вершинное k-расширение с количеством дополнительных ребер, равным:

$$k \sum_{j=1}^{|W^*|} \left(\left| W_j^* \right| + k \right)$$

Схема построения сводится к тому, чтобы провести из каждой вершины цвета $f(c)$, отличной от вершины c, ребра во все исходные и дополнительные вершины каждого из цветов W^*.

Пример построения МВ-2Р для цветной звезды, удовлетворяющей условиям следствия, представлен на рисунке 4.2.8.

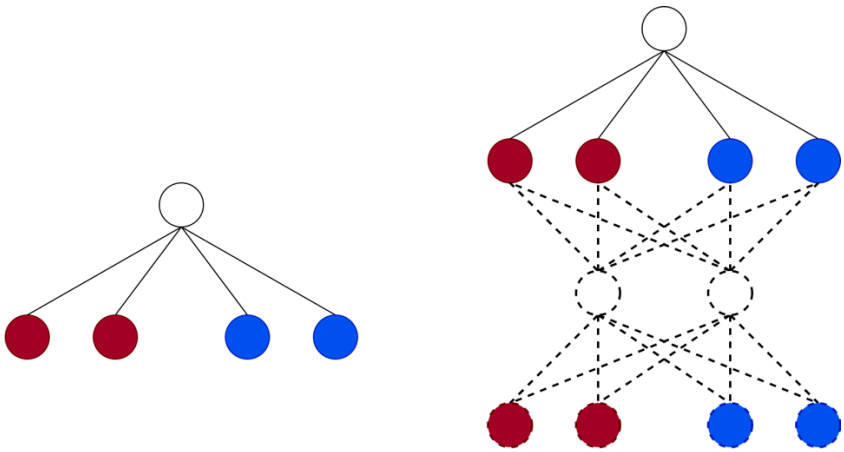

Рисунок 4.2.8 – Пример цветной звезды и ее МВ-2Р, построенной по схеме.

Теорема 4.2.3. Звезда $S = (V, \alpha, f)$ с условиями $|V_c| = 1$, $|W^1| > 1$, $|W^*| \geq 1$ имеет единственное с точностью до изоморфизма минимальное вершинное k-расширение с количеством дополнительных вершин, равным:

$$k \sum_{j=1}^{|W^*|} \left(\left| W_j^* \right| + k \right) + k(|W^1| - 1)$$

При этом схема построения МВ-kР будет следующей:

1. Для вершин, входящих в множество W^1, используется схема из теоремы 4.2.1. Соответственно, количество ребер будет равным $(|W^1| - 1)k$.

2. Для вершин из множеств, входящих в множество W^*, используется схема из теоремы 4.2.2 и ее следствия. Количество ребер будет равным $k\sum_{j=1}^{|W^*|}(|W_j^*|+k)$.

Пример цветного графа, удовлетворяющего условиям теоремы 4.2.3, и его МВ-1Р представлены на рисунке 4.2.9.

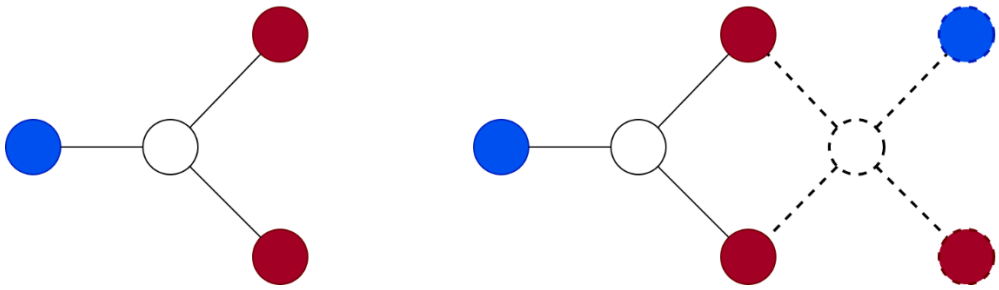

Рисунок 4.2.9 – Пример цветной звезды и ее МВ-1Р.

Следствие 4.2.2. Схема построения и количество дополнительных ребер из теоремы 4.2.3 справедливы для любых цветных звездных графов, в которых выполняется условие $|V_c|=1$, $|W^1|\geq 1$, $|W^*|\geq 0$.

Теперь перейдем к рассмотрению случая $|V_c|>1$. Для непомеченных звезд М.Б. Абросимов разработал и доказал схему построения минимальных вершинных k-расширений [1].

Теорема 4.2.4. Для звездного графа с количеством вершин, отличных от c, общее количество которых равно n, справедливы следующие утверждения:

1. При нечетном n и любом $k\in\mathbb{N}$ звездный граф имеет единственное с точностью до изоморфизма МВ-kР, равное K_k+S, где K_k – полный k-вершинный граф, а плюс – операция соединения. Подобные расширения называются тривиальными k-расширениями графа.

2. При четном n и четном k число МВ-kР в точности равно числу неизоморфных минимальных вершинных $(k-1)$-расширений данного графа, причем каждое из МВ-kР есть тривиальное 1-расширение соответствующего минимального вершинного $(k-1)$-расширения.

3. При четном n и нечетном k выделяются три случая:

3.1. При $k<n^2-2n$ звезда имеет единственное с точностью до изоморфизма МВ-kР – тривиальное k-расширение;

3.2. При $k = n^2 - 2n$ звезда имеет два с точностью до изоморфизма МВ-kР: тривиальное k-расширение и $R_{n+k+1,n+k+1}$.

3.3. При $k = n^2 - 2n$ звезда имеет единственное МВ-kР – граф $R_{n+k+1,n+k+1}$.

Оказывается, при использовании теорем 4.2.3 и 4.2.4 можно построить общую схему построения минимального вершинного k-расширения и для цветной звезды с $|V_c| > 1$.

Теорема 4.2.5. Для любого цветного звездного графа, где $|V_c| > 1, |W^1| \geq 1, |W^*| \geq 0$, минимальное вершинное k-расширение строится по следующей схеме:

1. Для вершин из V_c строится МВ-kР по теореме 4.2.4.

2. Для вершин из W^1 и $W^* \backslash \{V_c\}$ строится МВ-kР по следствию 4.2.2 теоремы 4.2.3.

Доказательство. Следует из теоремы 4.2.3 и ее следствия 4.2.2, а также из теоремы 4.2.4. ■

Пример цветного графа из теоремы 4.2.5 и его минимального вершинного k-расширения изображены на рисунке 4.2.10.

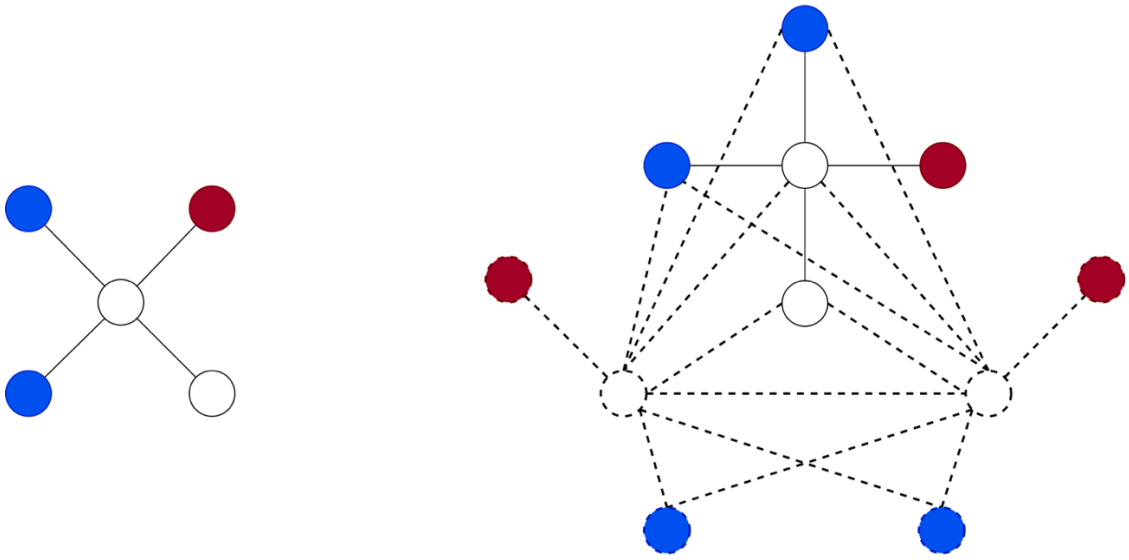

Рисунок 4.2.10 – Пример цветной звезды для теоремы 4.2.5 и ее МВ-2Р, построенного по указанной схеме.

Таким образом, теоремы 4.2.1–4.2.5 вместе со следствиями 4.2.1 и 4.2.2 предлагают схемы построения минимальных вершинных k-расширений для цветных неориентированных звездных графов всех возможных конфигураций.

Заключение

В диссертационной работе предложены и исследованы методы и алгоритмы построения технических систем различного типа, устойчивых к отказам элементов и связей между ними, а также всех соответствующих цветных реализаций и их минимальных вершинных и реберных k-расширений для заданного графа системы без проверки на изоморфизм. В результате исследования получены следующие теоретические и практические результаты.

1. Разработаны 7 алгоритмов, в том числе следующие алгоритмы: построение всех попарно-неизоморфных цветных реализаций для заданного простого графа, алгоритмы и построения всех неизоморфных минимальных вершинных и реберных k-расширений заданного цветного графа.

2. Проведены исследования разработанных алгоритмов, реализованы их параллельные версии, оценена их эффективность для различных видов графов. Эффективность алгоритмов варьируется в зависимости от видов графов и от формы, в которой графы подаются на вход.

3. Найдены схемы построения минимальных вершинных k-расширений цветных полных графов, а также схемы построения минимальных вершинных и реберных k-расширений цветных звезд.

Таким образом, все основные задачи диссертационного исследования решены. Документы, подтверждающие использование результатов диссертационного исследования, представлены в приложении к работе.

Список сокращений и условных обозначений

В-$\boldsymbol{k}$Р	вершинное $\boldsymbol{k}$-расширение
МВ-$\boldsymbol{k}$Р	минимальное вершинное k-расширение
МВ-Р, МВР	минимальное вершинное расширение
Р-$\boldsymbol{k}$Р	реберное k-расширение
МР-$\boldsymbol{k}$Р	минимальное реберное k-расширение
МР-Р, МРР	минимальное реберное расширение
$\boldsymbol{G = (V, \alpha)}$	неориентированный граф с множеством вершин V и отношением смежности α
$\boldsymbol{G = (V, \alpha, f)}$	неориентированный граф с множеством вершин V и отношением смежности α с введенной на нем функцией раскраски $f: V \to \{1, \dots, i\}$
$\boldsymbol{F = \{1, \dots, i\}}$	множество цветов графа
$\boldsymbol{i}$	количество уникальных цветов в графе
$\boldsymbol{M(G)}$	матрица смежности графа G
$\boldsymbol{d(v)}$	степень вершины v
$\boldsymbol{\{u, v\}}$	ребро между вершинами u и v
$\boldsymbol{K_n}$	полный n-вершинный граф
$\boldsymbol{S}$ или $\boldsymbol{K_{1,n}}$	звездный граф с одной вершиной степени n и с n вершинами степени 1
$\boldsymbol{R_{n,p}}$	регулярный n-вершинный граф, где каждая вершина имеет степень p
$\boldsymbol{C_n}$	циклический n-вершинный граф
$\boldsymbol{P_n}$	цепной n-вершинный граф
$\boldsymbol{O_n}$	вполне несвязный n-вершинный граф
$\boldsymbol{V_{f_j}}$	множество вершин графа, которым сопоставлен цвет f_i

$\boldsymbol{W}$	множество множеств вершин графа, которым сопоставлен некоторый цвет, для всех цветов из F
$\boldsymbol{W^1}$	множество, которое содержит все множества вершин с цветом, встречающимся лишь единожды в графе
$\boldsymbol{W^*}$	множество, которое содержит наборы вершин, сопоставленных с цветами, встречающимися в графе более одного раза
$\boldsymbol{v_{f_j}}$	вершина из исходного множества V, сопоставленная цвету f_i
$\boldsymbol{v^+_{f_j}}$	вершина из множества $V^*\backslash V$ дополнительных вершин, сопоставленная цвету f_i
$\boldsymbol{v_{W^1}}$	вершина v из множества V, которое включено в одно из множеств множества W^1
$\boldsymbol{v^+_{W^1}}$	вершина v из множества $V^*\backslash V$, которое включено в одно из множеств множества W^1
$\boldsymbol{v_{W^*}}$	вершина v из множества V, которое включено в одно из множеств множества W^*
$\boldsymbol{v^+_{W^*}}$	вершина v из множества $V^*\backslash V$, которое включено в одно из множеств множества W^*
Г-МК	алгоритм генерации неизоморфных цветных графов методом МакКея
Г-РФ	алгоритм генерации неизоморфных цветных графов методом Рида-Фараджева
Г-МК-р	алгоритм генерации неизоморфных графов с цветными ребрами методом МакКея
Г-РФ-р	алгоритм генерации неизоморфных графов с цветными ребрами методом Рида-Фараджева
Г-МВР	алгоритм генерации неизоморфных минимальных вершинных расширений цветного графа

Список литературы

1. Абросимов М.Б. *Графовые модели отказоустойчивости.* – Саратов: Издательство Саратовского университета, 2012. – 192 с.

2. Абросимов М.Б. *Минимальные расширения 4-, 5-, 6- и 7-вершинных графов*. – Сарат. гос. ун-т. – Саратов, 2000. – 26 с.; Деп. в ВИНИТИ 06.09.2000, № 2352-В00.

3. Абросимов М.Б. О сложности некоторых задач, связанных с расширениями графов // *Математические заметки.* – 2010. – № 88:5. – С. 643–650.

4. Абросимов М.Б., Бондаренко П.П. Исследование минимальных вершинных и реберных 1-расширений цепей с вершинами двух типов // Свид. о гос. регистрации программы для ЭВМ № 2010616497, выданное Роспатентом. Зарегистрирована в Реестре программ для ЭВМ 30.09.2010.

5. Абросимов М.Б., Бондаренко П.П. О минимальных вершинных 1-расширениях циклов с вершинами двух типов // *Прикладная дискретная математика.* – 2011. – № 4. – С. 80–81.

6. Абросимов М.Б., Долгов А.А. О реконструируемости малых турниров // *Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика.* – 2009. – Т. 9, вып. 2. – С. 94–98.

7. Абросимов М.Б., Долгов А.А. Семейства точных расширений турниров // *Прикладная дискретная математика.* – 2008. – № 1. – С. 101–107.

8. Абросимов М.Б., Камил И.А.К. Разработка системы предотвращения вторжений с использованием параллельного программирования и системы отказоустойчивости // *Безопасность информационных технологий.* – 2018. – Т. 25, № 1. – С. 65–73.

9. Абросимов М.Б., Камил И.А.К., Лобов А.А. Построение всех неизоморфных минимальных вершинных расширений графа методом канонических

представителей // *Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика.* – 2019. – Т. 19, вып. 4. – С. 479–485.

10. Абросимов М.Б., Камил И.А.К., Судани Х.Х.К., Лобов А.А. Построение оптимальных отказоустойчивых реализаций графов FTConstructor // Свид. о гос. регистрации программы для ЭВМ № 2020614773. Зарегистрирована в Реестре программ для ЭВМ 24.04.2020.

11. Абросимов М.Б., Моденова О.В. Характеризация орграфов с малым числом дополнительных дуг минимального вершинного 1-расширения // *Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика*. – 2013. – Т. 13., вып. 2, ч. 2. – С. 3–9.

12. Абросимов М.Б., Моденова О.В. Характеризация орграфов с тремя дополнительными дугами в минимальном вершинном 1-расширении // *Прикладная дискретная математика*. – 2013. – № 3. – С. 68–75.

13. Абросимов М.Б., Разумовский П.В. Генерация неизоморфных вершинных k-раскрасок графа // *Компьютерные науки и информационные технологии: материалы Междунар. науч. конф.* – Саратов: центр «Наука», 2016. – С. 13–15.

14. Абросимов М.Б., Разумовский П.В. О генерации неизоморфных раскрасок методом Рида-Фараджева // *ПДМ. Приложение.* – 2019. – № 12. – С. 173–176.

15. Абросимов М.Б., Судани Х.Х.К., Лобов А.А. Построение минимальных рёберных расширений графа без проверки на изоморфизм // *Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика.* – 2020. – Т. 20, вып. 1. – С. 105–115.

16. Арлазаров В.Л., Зуев И.М., Усков А.В., Фараджев И.А. Алгоритм приведения неориентированных графов к каноническому виду // *Ж. вычисл. матем. и матем. физ.* – 1974. – Т. 14, № 3. – С. 737–743.

17. Богомолов А.М., Салий В.Н. *Алгебраические основы теории дискретных систем*. – М.: Наука, 1997. – 368 с.

18. Гавриков А.В. Т-неприводимые расширения объединений некоторых

типов орграфов // *Прикладная дискретная математика*. – № 4 (22). – 2013. – С. 47–56.

19. Долгов А.А. О семействах точных вершинных k-расширений графов при k > 1 // *Материалы Международного молодежного научного форума «Ломоносов-2010»*. – М.: МАКС Пресс, 2010. – С. 30–32.

20. Долгов А. А. Семейство точных 2-расширений турниров // *Прикладная дискретная математика*. – 2010. – № 9. – С. 96–99.

21. Зыков А.А. О некоторых свойствах линейных комплексов // *Математический сборник.* – 1949. – Т. 24(66), № 2. – С. 163–188.

22. Кабанов М. А. Об отказоустойчивых реализациях графов // *Теоретические задачи информатики и ее приложений*. – Саратов: Изд-во Сарат. ун-та, 1997. – Вып.1. – С. 50–58.

23. Камил И.А.К. Обеспечение отказоустойчивости высокопроизводительного узла параллельных вычислений с интерфейсом передачи сообщений (MPI) // *Современная наука: актуальные проблемы теории и практики: Серия «Естественные и Технические науки»*. – 2018. – № 4. – С. 57–59.

24. Камил И.А.К., Абросимов М.Б. Разработка системы предотвращения вторжений с использованием параллельного программирования и технологий отказоустойчивости // *Технические средства защиты информации: Тезисы докладов XVI Белорусско-российской научно-технической конференции*, 5 июня 2018 г., Минск. Минск: БГУИР, 2018. – С. 44.

25. Камил И.А.К., Абросимов М.Б., Лобов А.А. Построение минимальных вершинных расширений графа методом Рида-Фараджева // *International Journal of Open Information Technologies.* – 2020. – Vol. 8, no. 4. – P. 54–58.

26. Камил И.А.К., Судани Х.Х.К., Абросимов М.Б. К вопросу о параллельных алгоритмах построения минимальных вершинных и реберных 1-расширений графов // *Компьютерные науки и информационные технологии: Материалы Междунар. науч. конф.* – Саратов: Издат. центр «Наука», 2018. – С. 173–176.

27. Камил И.А.К., Судани Х.Х.К., Лобов А.А., Абросимов М.Б. Построение минимальных расширений графа методом канонических представителей // *Прикладная дискретная математика. Приложение.* – 2019. – № 12. – С. 179–182.

28. Каравай М.Ф. Инвариантно-групповой подход к исследованию *k*-отказоустойчивых структур // *Автоматика и телемеханика*. – 2000. – № 1. – С. 144–156.

29. Каравай М.Ф. Минимизированное вложение произвольных гамильтоновых графов в отказоустойчивый граф и реконфигурация при отказах. I. Одноотказоустойчивые структуры // *Автоматика и телемеханика*. – 2004. – № 12. – С. 159–177.

30. Каравай М.Ф. Минимизированное вложение произвольных гамильтоновых графов в отказоустойчивый граф и реконфигурация при отказах. II. Решетки и k-отказоустойчивость // *Автоматика и телемеханика*. – 2005. – № 2. – С. 175–189.

31. Каравай М.Ф. Применение теории симметрии к анализу и синтезу отказоустойчивых систем // *Автоматика и телемеханика*. – 1996. – № 6. – С. 159–173.

32. Киреева А.В. Отказоустойчивость в функциональных графах // *Упорядоченные множества и решетки*. – Саратов: Изд-во Сарат. ун-та, 1995. – Вып. 11. – С. 32–38.

33. Курносова С.Г. Т-неприводимые расширения объединений полных графов // *Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика*. – 2005. – Т. 5, вып. 1. – С. 107–115.

34. Курносова С.Г. Т-неприводимые расширения полных бинарных деревьев // *Вестн. Томск. гос. ун-та. Приложение*. – 2005. – № 14. – С. 158–160.

35. Павлов Д.А. Каноническая нумерация графов и библиотека nauty // *КИО, раздел «Информатика»,* № 5, 2009. [Электронный ресурс] – URL: https://cyberleninka.ru/article/n/kanonicheskaya-numeratsiya-grafov-i-biblioteka-nauty.

36. Разумовский П.В., Абросимов М.Б. Построение цветных графов без проверки на изоморфизм // *Изв. Сарат. ун-та. Нов. Сер. Сер. Математика. Механика. Информатика*. – 2021. – Т. 21, вып. 2. – С. 267–277.

37. Разумовский П.В., Абросимов М.Б. *ColorGraphExtensions* // Свидетельство о государственной регистрации программы для ЭВМ № 2021662022, выданное Роспатентом. Зарегистрировано в Реестре программ для ЭВМ 20.06.2021.

38. Сухов С.А. *DSR Generator* // Свид. о гос. регистрации программы для ЭВМ № 2016610073. Зарегистрирована в Реестре программ для ЭВМ 11 января 2016 г.

39. Харари Ф. *Теория графов.* – М.: Мир, 1973. – 296 с.

40. Харченко В.С. Гарантоспособность и гарантоспособные системы: элементы методологии // *Радіоелектроні и комп'ютерні системи*. – 2006. - № 5. – С. 7–19.

41. Харченко В.С. Парадигмы и принципы гарантоспособных вычислений: состояние и перспективы развития // *Радіоелектроні и комп'ютерні системи*. – 2009. – № 2(36). – С. 91–100.

42. Фараджев И.А. (ред.) *Алгоритмические исследования в комбинаторике*. – М.: Наука, 1978. – 190 с.

43. About Fugaku [Электронный ресурс] // *RIKEN Center for Computational Science*. – URL: https://www.r-ccs.riken.jp/en/fugaku/about/ (дата обращения: 15.10.2021).

44. Avižienis A. Design of fault-tolerant computers // *AFIPS'67 Conf. Proc*. – New York: ACM, 1967. – P. 733–743.

45. Avižienis A. Fault-Tolerant Computing: An Overview // *IEEE Computer*. – 1971. – Vol. 4, № 1. – P. 5–8.

46. Aviženis A. Fault-tolerance and fault-intolerance: Complementary approaches to reliable computing // *Proc. Intern. Conf. of Reliable Software,* New York, 1975. – P. 458–464.

47. Aviženis A., Laprie J.-C., Randell B., Landwehr C. Basic Concepts and Taxonomy of Dependable and Secure Computing // *IEEE Trans. on Dependable and Secure Comput.* – 2004. – № 1. – P. 11–33.

48. Berge C. Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind // *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe.* – 1961. – Vol. 10. – P. 114.

49. Brinkmann G. Fast generation of cubic graphs // *J. Graph Theory*. – 1996. – Vol. 23, № 2. – P. 139–149.

50. Brinkmann G. Isomorphism rejection in structure generation programs // *Discrete Mathematical Chemistry, DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* – 2000. – Vol. 51. – P. 25–38.

51. Brinkmann G., Goedgebeur J. Generation of cubic graphs and snarks with large girth // *Journal of Graph Theory*. – 2017. – Vol. 86, № 2. – P. 255–272.

52. Brinkmann G., Goedgebeur J., Hagglund J., Markstrom K. Generation and properties of Snarks // *Journal of Combinatorial Theory, Series B*. – 2013. – Vol. 103, № 4. – P. 468–488.

53. Brinkmann G., Goedgebeur J., McKay B. D. Generation of cubic graphs // *Discrete Math. Theor. Comput. Sci*. – 2011. – Vol. 13, № 2. – P. 69–80.

54. Bron C., Kerbosh J. Algorithm 457 – Finding all cliques of an undirected graph // *Comm. of ACM*. – 1973. – Vol. 16. – P. 575–577.

55. Bruck J., Cypher R., Ho C. Fault-tolerant meshes with small degree // *SIAM J. Comput*. – 1997. – Vol. 26, № 6. – P. 1764–1784.

56. Carter W.C., Bouricius W.G. A Survey of Fault Tolerant Computer Architecture and its Evaluation // *IEEE Computer*. – 1971. – Vol. 4, № 1. – P. 9–16.

57. Cayley A. On the colourings of maps // *Proceedings of the Royal Geographical Society, Blackwell Publishing*. – 1879. – Vol. 1, no. 4. – P. 259–261.

58. Chaitin G.J. Register allocation & spilling via graph colouring // *Proc. 1982 SIGPLAN Symposium on Compiler Construction.* – 1982. – P. 98–105.

59. Chou R.S., Hsu L.H. 1-edge fault-tolerant designs for meshes // *Parallel Process. Lett*. – 1994. – Vol. 4, № 4. – P. 385–389.

60. Choudum S. A., Sivagurunathan S. Optimal fault-tolerant networks with a server // *Networks*. – 2000. – Vol. 35, № 2. – P. 157–160.

61. Colburn C.J., Read R.C. Orderly algorithms for generating restricted classes of graphs // *Journal of Graph Theory*. – 1979. – Vol. 3. – P. 187–195.

62. Colburn C.J., Read R.C. Orderly algorithms for graph generation // *Intern. J. Computer Math*. – 1979. – Sec. A.7. – P. 167–172.

63. Dailey D.P. Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete // *Discrete Mathematics*. – 1980. – Vol. 30, no. 3. – P. 289–293.

64. Darga P.T., Liffiton M.H., Sakallah K.A., Markov I.L. Exploiting structure in symmetry detection for CNF // *Proceedings of the 41st Design Automation Conference*, 2004. – P. 530–534.

65. Duncan L.R., Perry A.D. A method of matrix analysis of group structure // *Psychometrika*. – 1949. – Vol. 2, no. 14. – P. 95–116.

66. Dutt S., Hayes J.P. Designing fault-tolerant systems using graph automorphisms // *Journal of Parallel and Distributed Computing*. – 1991. – Vol. 12. – P. 249–268.

67. Encyclopedia of Parallel Computing // *ed. D. A. Padua*. – New York: Springer. – 2011.

68. Erlebach T., Jansen K. The complexity of path coloring and call scheduling // *Theoretical Computer Science*. – 2001. – Vol. 255, is. 1-2. – P. 33-50. [ERLEBACH]

69. Faradzhev I.A. Constructive enumeration of combinatorial objects // *Colloques internationaux C.N.R.S. №260, Problemes Combinatoires et Theorie des Graphes*, Orsay. – 1976. – P. 131–135.

70. Farrag A.A., Dawson R.J. Designing optimal fault-tolerant star networks // *Networks*. – 1989. – Vol. 19. – P. 707–716.

71. Gandham S., Dawande M., Prakash R. Link scheduling in sensor networks: distributed edge coloring revisited // *Proc. IEEE 24th Annual Joint Conf. of the IEEE Comp. and Communications Soc*. – 2005. – Vol. 4. – P. 2492–2501.

72. Garey M.R., Johnson D.S. *Computers and Intractability: a guide to the theory of NP-Completeness*. – New York: Freeman, 1985. – 338 p.

73. Garey M.R., Johnson D.S., Stockmeyer L. Some simplified NP-complete problems // *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing.* – 1974. – P. 47–63.

74. Gonzalez T., Sahni S. Open shop scheduling to minimize finish time // *Journal of ACM.* – 1976. – Vol. 23, no. 4. – P. 665–679.

75. Grund R. Konstruktion schlichter Graphen mit gegebener Gradpartition // *Bayreuther Mathematische Schriften*. 1993. Vol.44. P.73–104.

76. Gurevich Y. From invariants to canonization // *Bulletin of the European Association of Theoretical Computer Science.* – 1997. – Vol. 63. – P. 115–119.

77. Harary F., Hayes J.P. Edge fault tolerance in graphs // *Networks.* – 1993. – Vol. 23. – P. 135–142.

78. Harary F., Hayes J.P. Node fault tolerance in graphs // *Networks.* – 1996. – Vol. 27. – P. 19–23.

79. Hayes J.P. A graph model for fault-tolerant computing system // *IEEE Trans. Comput.* – 1976. – Vol. C-25, no. 9. – P. 875–884.

80. Hsu L. H., Lin C. K. *Graph Theory and Interconnection Networks*. – New York: CRC Press, 2009.

81. Junttila T., Kaski P. Engineering an efficient canonical labeling tool for large and sparse graphs // *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments and the 4th Workshop on Analytic Algorithms and Combinatorics*, 2007. – P. 135–149.

82. Junttila T., Kaski P. Conflict Propagation and Component Recursion for Canonical Labeling // *Proceedings of the 1st International ICST Conference on Theory and Practice of Algorithms*, 2011. P. 151–162.

83. Karp R.M. Reducibility among combinatorial problems // *Complexity of Computer Computations*. – 1972. – New York: Plenum. – P. 85–103.

84. Laprie J.-C. *Dependability: Basic Concepts and Terminology*, 1992. – NY: Springer-Verlag. – 245 p.

85. Laprie J.-C. Dependable Computing and Fault Tolerance: Concepts and Terminology // *Proc. 15th IEEE Intern. Symp. Fault-Tolerant Computing (FTCS-15).* – 1985. – NY: ACM. – P. 2–11.

86. Lopez-Presa J.L., Fernandez Anta A. Fast algorithm for graph isomorphism testing // *Proceedings of the 8th International Symposium on Experimental Algorithms*, 2009. – P. 221–232.

87. Lopez-Presa J.L., Fernandez Anta A., Nunez Chiroque L. Conauto-2.0: Fast isomorphism testing and automorphism group computation, 2011. [Электронный ресурс]. – URL: http://arxiv.org/abs/1108.1060.

88. McKay B.D. Computing automorphisms and canonical labellings of graphs // *Combinatorial Mathematics, Lecture Notes in Mathematics*. – 2006. – Berlin: Springer-Verlag. – P. 223–232.

89. McKay B.D. Description of graph6, sparse6 and digraph6 encodings [Электронный ресурс]. – URL: http://cs.anu.edu.au/people/bdm/data/formats.txt (дата обращения: 14.09.2021).

90. McKay B.D. *Combinatorial Data: Graphs*. [Электронный ресурс]. – URL: https://users.cecs.anu.edu.au/~bdm/data/graphs.html.

91. McKay B.D. Isomorphism-free exhaustive generation // *Journal of Algorithms.* – 1998. – Vol. 26. – P. 306–324.

92. McKay B.D. *nauty* User's Guide (version 1.5) // *Tech. Rpt. TR-CS-90-02, Dept. Computer Science*. – 1990. – Australia: Austral. Nat. Univ.

93. McKay B.D. Practical graph isomorphism // *Congr. Numer.* – 1980. – Vol. 30. – P. 45–87.

94. McKay B.D., Piperno A. Practical Graph Isomorphism, II // *Journal of Symbolic Computation.* – 2014. – Vol. 60. – P. 94–112.

95. Meringer M. Fast generation of regular graphs and construction of cages // *Journal of Graph Theory*. 1999. Vol.30. P.137–146.

96. Molloy M., Reed B. *Graph colouring and the probabilistic method*. – Berlin: Springer, 2002. – 326 p.

97. Read R.C. Every one a Winner or how to Avoid Isomorphism Search when Cataloguing Combinatorial Configurations // *Annals of Discrete Mathematics*. – 1978. – Vol. 2. – P. 107–120.

98. Read R.C., Corneil D.G. The graph isomorphism disease // *J. Graph Theory1*. – 1977. – P. 339–363.

99. Sung T. Y., Ho T. Y., Chang C. P., Hsu L. H. Optimal k-fault-tolerance network for token rings // *J. Inform. Science and Engineering*. – 2000. – № 16. – P. 381–390.

100. TOP500 [Электронный ресурс] // *TOP500*. – URL: https://www.top500.org/ (дата обращения: 15.10.2021).

101. von Neumann J. Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components // *Automata Studies, ser. Annals of Mathematics Studies. Princeton*, NJ: Princeton University Press, 1956. – Vol. 34. – P. 43–98.

102. Weygant P. *Clusters for high availability: a primer of HP solutions*, 2001. – NJ: Prentice Hall. – 296 p.

103. Whitney H. Congruent graphs and the connectivity of graphs // *American Journal of Mathematics.* – 1932. – Vol. 54, no. 1. – P. 160–168.

104. World’s fastest supercomputer Fugaku begins its shared use [Электронный ресурс] // *Science | Business*. – URL: https://sciencebusiness.net/network-updates/worlds-fastest-supercomputer-fugaku-begins-its-shared-use (Дата обращения: 15.10.2021).

Приложение А

Свидетельство о государственной регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2021662022

ColorGraphExtensions

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Саратовский национальный исследовательский государственный университет имени Н.Г. Чернышевского» (RU)***

Авторы: ***Разумовский Пётр Владимирович (RU), Абросимов Михаил Борисович (RU)***

Заявка № **2021661289**
Дата поступления **20 июля 2021 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***20 июля 2021 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев