Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

«Нарушение симметрий в графах на основе обхода в ширину»

Автор: Моклев Вячеслав Владимирович _____
Направление подготовки (специальность): 01.03.02 Прикладная математика и информатика
Квалификация: Бакалавр
Руководитель: Ульянцев В.И., канд. техн. наук. _____

К защите допустить
Зав. кафедрой Васильев В.Н., докт. техн. наук, проф. _____

«___» _____ 20___ г.

Санкт-Петербург, 2017 г.

Студент Моклев В.В. **Группа** М3439 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования

Направленность (профиль), специализация Математические модели и алгоритмы разработки программного обеспечения

Квалификационная работа выполнена с оценкой _____

Дата защиты «___» _____ 20___ г.

Секретарь ГЭК _____ Принято: «___» _____ 20___ г.

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

АННОТАЦИЯ
ПО ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Студента Моклев В. В.
(Фамилия, И., О.)

Факультет Информационных технологий и программирования

Кафедра Компьютерных технологий **Группа** M3439

Направление подготовки Прикладная математика и информатика

Квалификация(степень) Бакалавр прикладной математики и информатики

Наименование темы: Нарушение симметрий в графах на основе обхода в ширину

Научный руководитель Ульянцев В. И., к.т.н., доц. каф. КТ
(Фамилия, И., О., ученое звание, степень)

КРАТКОЕ СОДЕРЖАНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ И ОСНОВНЫЕ ВЫВОДЫ

объем _____ стр., **графический материал** 0 стр., **библиография** _____ наим.

- Направление и задача исследований

Задачей исследования является разработка методов нарушения симметрии для задач поиска графов на основе обхода графов в ширину.

- Проектная или исследовательская часть (с указанием основных методов исследований, расчетов и результатов)

В рамках работы были разработаны методы нарушения симметрии для задач поиска графов. Также были предложены формальные доказательства корректности всех предложенных методов.
Реализованы предложенные методы и применены к нескольким задач поиска графов.

- Экономическая часть (какие использованы методики, экономическая эффективность результатов)

Не рассматривалась

- Характеристика вопросов экологии, техники безопасности и др.

Не рассматривалась

- Является ли работа продолжением курсовых проектов (работ), есть ли публикации

Работа является продолжением курсового проекта по производственной практике. Данная работа была представлена в рамках VI Всероссийского конгресса молодых ученых и была отмечена дипломом за лучший научно-исследовательский доклад.

Практическая ценность работы. Рекомендации по внедрению

Результаты, полученные в данной работе, могут быть использованы во многих областях, где возникает потребность в решении задачи графового поиска. Также они могут быть использованы для ускорения уже существующих система поиска графов, построенных на основе сведения к задачам SAT или CSP.

Выпускник_____
(подпись)

Научный руководитель_____
(подпись)

“_____” _____ 20_____ г.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет *Информационных Технологий и Программирования*
Кафедра *Компьютерных технологий* **Группа** *M3439*
Направление (специальность) *Прикладная математика и информатика*
Квалификация (степень) *Бакалавр прикладной математики и информатики*

З А Д А Н И Е
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студент *Моклев В.В.*
Руководитель *Ульянцев В.И., канд. техн. наук, доцент кафедры КТ Университета ИТМО*

1. Наименование темы *Нарушение симметрий в графах на основе обхода в ширину*

2. Срок сдачи студентом законченной работы *31 мая 2017 г.*

3. Техническое задание и исходные данные к работе
В рамках работы требуется разработать методы нарушения симметрии для задачи поиска неориентированных графов, основанные на обходе графа в ширину. Требуется изучить существующие методы нарушения симметрии в графах и автоматах, актуальные задачи поиска графов, средства решения задач SAT и CSP, а также провести сравнение разработанных методов с существующими на нескольких актуальных задачах поиска графов.

4. Содержание выпускной работы (перечень подлежащих разработке вопросов)
а) Обзор предметной области
б) Разработка методов нарушения симметрии для задачи поиска графов
в) Реализация предложенных и существующих методов нарушения симметрии
г) Проведение экспериментов и сравнение результатов

5. Перечень графического материала (с указанием обязательного материала)
Не предусмотрено

6. Исходные материалы и пособия

1. *Ulyantsev V., Zakirzyanov I., Shalyto A. BFS-based symmetry breaking predicates for DFA identification // International Conference on Language and Automata Theory and Applications. – Springer International Publishing, 2015. – С. 611-622.*

2. *Codish M. et al. Breaking Symmetries in Graph Representation // IJCAI. – 2013. – С. 3-9.*

3. *Biere A., Heule M., van Maaren H. (ed.). Handbook of satisfiability. – IOS press, 2009. – Т. 185.*

7. Консультанты по работе с указанием относящихся к ним разделов работы

Раздел	Консультант	Подпись, дата	
		Задание выдал	Задание принял
Экономика и организация производства			
Технология приборостроения			
Безопасность жизнедеятельности и экология			

КАЛЕНДАРНЫЙ ПЛАН

№№ п/п	Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Примечание
1	Ознакомление с предметной областью	20.11.2016	
2	Адаптация автоматного метода для графов	15.02.2017	
3	Поиск актуальных задач	30.02.2017	
4	Экспериментальное сравнение, анализ результатов	05.03.2017	
5	Построение более строгих предикатов	01.04.2017	
6	Реализация всех описанных методов	20.04.2017	
7	Экспериментальное сравнение	01.05.2017	
8	Написание пояснительной записки	30.05.2017	

8. Дата выдачи задания _____ *1 сентября 2016 г.* _____

Руководитель _____

Задание принял к исполнению _____

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Теория графов — один из обширных разделов информатики, находящий применение повсюду. Задачи, связанные с графами, возникают в теории управления, логистике, медицине, биологии, теории формальных языков и множестве других областей. Для многих графовых задач разработаны эффективные алгоритмы их решения, но для многих задач таких алгоритмов не существует, либо они еще не были открыты.

Одним из видов таких задач являются задачи по поиску графов, обладающих некоторыми свойствами. Для большого класса таких свойств данные задачи являются NP-полными, NP-трудными, или не известны алгоритмы, которые решают данные задачи за полиномиальное время.

Распространенной техникой решения задач, у которых не известно полиномиального алгоритма решения, является сведение к задаче SAT (Boolean SATisfiability Problem, задача выполнимости булевой формулы) или CSP (Constraint Satisfaction Problem, задача удовлетворимости ограничений). Для многих задач существует полиномиальное сведение к этим задачам, а если исходная задача принадлежала к классу NP, то такое сведение существует всегда.

Для решения задач SAT и CSP существует множество специальных программных средств. Несмотря на то, что неизвестен полиномиальный алгоритм решения этих задач, современные программные средства решения задач выполнимости являются очень мощным инструментом и позволяют решать задачи огромных размеров (с большим количеством переменных и ограничений). Также идет активное развитие параллельных средств решения SAT, которые могут использовать все доступные им ядра или процессоры для решения поставленной задачи. Каждый год проходит соревнование SAT Competition [1], в котором соревнуются в производительности на разных классах задач множество таких программных средств.

Однако для многих задач сведение к SAT или CSP для каждого решения исходной задачи порождает множество решений полученной задачи. Такие решения обычно называются изоморфными и составляют целый класс изоморфизма, соответствующий одному решению исходной задачи. Большое количество таких решений существенно увеличи-

вают пространство поиска и время решения задачи. Для правильного решения задачи было бы достаточно проверять только одно решение из каждого класса изоморфизма.

Для уменьшения количества изоморфных решений применяются техники нарушения симметрии, например [2] и [3]. Такие техники широко изучаются и предложено множество способов эффективно избавляться от некоторых или всех изоморфных решений задачи в разных ситуациях.

В данной работе предлагаются такие нарушения симметрии для неориентированных графов и их применение к реальным задачам поиска графов.

Методы, предлагаемые в данной работе, основаны на предложенной в работе [4] идее использовать порядок обхода BFS. Далее предлагаются последовательные улучшения метода, основанного на порядке BFS, которые устраняют больше симметрий в графе путем ограничения выбора стартовой вершины обхода и порядка обхода ребер в графе. Предложенные методы учитывают внутреннюю структуру графа для более эффективного устранения симметрий в графе. Однако эти методы подходят только для поиска связных графов, поэтому далее в работе предлагает обобщение этих методов на случай произвольных графов.

В конце работы проводится сравнение предложенных методов нарушения симметрии с существующими на данный момент на примере нескольких задач из экстремальной теории графов.

ГЛАВА 1. ОБЗОР МЕТОДОВ НАРУШЕНИЯ СИММЕТРИИ

В данной главе вводятся основные определения, которые понадобятся в дальнейшей работе, рассматривается решаемая задача, существующие на данный момент методы, а также приводятся некоторые актуальные и известные постановки решаемых задач.

1.1. Основные определения

В данном разделе вводятся определения, которые будут встречаться в работе в дальшейшем.

Определение 1. (Занумерованный) неориентированный граф — упорядоченная пара $\langle V, E\rangle$, где V — некоторое конечное множество меток (далее в работе всегда $V = \{0\,..\,n-1\}$), $E \subset V \times V$ — множество ребер.

Определение 2. Графы G_1 и G_2 называются изоморфными, если существует перестановка вершин первого графа такая, что в результате этой перестановки получается второй граф: $\exists\pi\colon \pi(E(G_1)) = E(G_2)$.

Отношение изоморфизма обозначается $(\sim)$: $G_1 \sim G_2$.

Определение 3. Пусть $\mathcal{G}_n$ — множество занумерованных неориентированных графов из n вершин. Множеством незанумерованных графов будет называть факторизацию этого множества по отношению изоморфизма: $\tilde{\mathcal{G}}_n = \mathcal{G}_n/\!\sim$.

Незанумерованный неориентированный граф — элемент этого множества, представляет собой класс изоморфизма целиком, это любой граф из класса изоморфизма, у которого убрали все метки с вершин: $[G] = G/\!\sim, [G] \in \tilde{\mathcal{G}}_n, G \in \mathcal{G}_n$.

Определение 4. Предикат — отображение из множества объектов в множество булевых значений: $\varphi\colon X \to \mathcal{B}$, где $\mathcal{B} = \{\text{false}, \text{true}\}$.

Определение 5. Матрица смежности A графа $G \in \mathcal{G}_n$ — матрица размера $n \times n$, состоящая из булевых элементов ($A \in \mathcal{B}^{n\times n}$), такая, что $A[u, v] = \text{true} \Leftrightarrow (u, v) \in E(G)$.

Определение 6. Порядок обхода в ширину (BFS-нумерация) — такой порядок вершин, в котором их посещает алгоритм поиска в ширину (BFS, breadth first search) при старте в вершине с номером 0.

Определение 7. Дерево BFS графа G — дерево с вершинами $V(G)$, корнем v_0 и ребрами, полученными при обходе BFS — ребро (u, v) соеди-

няет вершины в дереве, если вершина v была добавлена в очередь при открытии вершины u.

Определение 8. Обхватом графа называется длина кратчайшего цикла в нем.

Определение 9. Хроматическим числом графа $\chi(G)$, $G \in \mathcal{G}_n$ называется минимальное количество цветов, в которое можно раскрасить вершины графа G таким образом, чтобы никакие две вершины, соединенные ребром, не были покрашены в одинаковый цвет.

Также определим обозначения для нескольких самых известных семейств графов.

Определение 10. Полный граф $K_n \in \mathcal{G}_n$ — граф, в котором любая пара вершин соединена ребром.

Определение 11. Путь $P_n \in \mathcal{G}_n$ — граф, в котором $n-1$ ребро и все вершины выстроены в «цепочку», то есть $E(P_n) = \{\langle 0,1\rangle, \langle 1,2\rangle, \ldots, \langle n-2, n-1\rangle\}$.

Определение 12. Цикл $C_n \in \mathcal{G}_n$ — граф, который получен из пути P_n добавлением еще одного ребра $\langle n-1, 0\rangle$, то есть $E(C_n) = E(P_n) \cup \langle n-1, 0\rangle$.

Определение 13. Полный двудольный граф $K_{n,m} \in \mathcal{G}_{n+m}$ — граф, в котором все вершины можно разделить на две «доли»: $V = V_1 \cup V_2$, такие, что из каждой вершины первой доли ведут ребра во все вершины второй доли: $(u,v) \in E(K_{n,m}) \Leftrightarrow (u \in V_1 \wedge v \in V_2) \vee (u \in V_2 \wedge v \in V_1)$.

Определение 14. Звезда $S_n \in \mathcal{G}_{n+1}$ — граф $K_{1,n}$, который представляет собой вершину степени n, которая соединена со всеми остальными вершинами, каждая из которых имеет степень 1.

1.2. Постановка задачи

Задача существования графа заключается в поиске графа с заданным количеством вершин, который удовлетворяет заданным свойствам. Примером такой задачи является следующая: найти граф из n вершин и m ребер, который не содержит циклов длины 3 и 4. Некоторые виды ограничений порождают задачу поиска графа, у которой известно полиномиальное решение, однако в общем случае это не так — либо полиномиального решения не существует в принципе, либо оно неизвестно, а его разработка может занять неограниченное количество времени.

Для таких задач в последнее время все бóльшую популярность набирают методы решения, основанные на сведении задачи к SAT или CSP. Популярность таких методов заключается в том, что с каждым годом растет производительность средств решения задач выполнимости (SAT и CSP), позволяя решать другие задачи путем сведения к задаче выполнимости за допустимое время, в некоторых случаях даже быстрее, чем с помощью алгоритмов, разработанных специально для решения конкретной задачи.

Стандартная схема сведения задачи поиска графа к задаче SAT или CSP состоит из трех этапов. Сначала в модели CSP заводятся переменные, которые отвечают за элементы матрицы смежности (n^2 или $\frac{n(n-1)}{2}$ булевых переменных $A_{i,j}$), и на эти переменные накладываются условия, которые гарантируют корректность найденной матрицы смежности (симметричность, отсутствие петель). Затем вводятся дополнительные переменные и ограничения, которые задают исходное ограничение из задачи поиска графа. И на последнем этапе к модели добавляются дополнительные переменные и ограничения, которые являются следствиями из уже имеющихся в модели ограничений, однако в некоторых ситуациях могут дать больше информации программному средству, с помощью которого будет осуществляться нахождение решения модели.

Однако, у сведения к задачам выполнимости есть существенный недостаток. Большинство ограничений, которые формулируются для задач поиска графов, формулируют незанумерованными графами. Однако, большинство сведений графовых задач к задаче выполнимости в явном виде вводят нумерацию вершин, таким образом для каждого решения исходной задачи построенная модель задачи выполнимости будет иметь множество решений, которые соответствуют целому классу изоморфизма исходного графа. Это существенно увеличивает время нахождения решения, из-за чего сведение к задаче выполнимости может существенно проигрывать другим методам.

Для решения такой проблемы применяются методы нарушения симметрии. Такие методы заключатся в том, что к ограничениям, полученным из задачи, добавляются дополнительные ограничения (предикаты нарушения симметрии), цель которых — запретить как можно

больше графов из каждого класса изоморфизма. Однако такие ограничения должны оставлять хотя бы одного представителя из каждого класса изоморфизма, в противном случае может быть запрещен целый класс изоморфизма, который соответствовал решению исходной задачи, таким образом будет потеряно решение.

Определение 15. Предикат нарушения симметрии для задачи поиска графов $\varphi(G)$ является корректным, если для любого графа найдется изоморфный ему, который допускается этим предикатом: $\forall G \in \mathcal{G}_n \; \exists G' \sim G \colon \varphi(G') = \text{true}$.

Однако, в некоторых случаях возможно использовать предикаты нарушения симметрии, которые не являются корректными. Если заранее известно, что решение поставленной задача поиска графов лежит в некотором классе графов, то если предикат нарушения симметрии будет запрещать все графы вне заданного класса, то ни одно решение исходной задачи не будет потеряно в результате нарушения симметрий.

Определение 16. Предикат нарушения симметрии для задачи поиска графов $\varphi(G)$ является корректным для некоторого множества графов $\mathcal{G}$, если для любого графа из множества $\mathcal{G}$ найдется изоморфный ему, который допускается этим предикатом: $\forall G \in \mathcal{G} \; \exists G' \sim G \colon \varphi(G') = \text{true}$.

В качестве примера такого класса графов можно привести множество связных графов. В данной работе предложены предикаты нарушения симметрии, являющиеся корректными для множества связных графов. Такие предикаты можно применять для любой задачи поиска графов, в которой ответом всегда является связный граф.

1.3. Существующие методы

В данном разделе излагаются результаты, полученные в ранее опубликованных работах в области нарушения симметрии в графах.

1.3.1. Каноническое представление

Одним из методов проверки графов на изоморфизм является каноническое представление графа.

Определение 17. Пусть $G \in \mathcal{G}_n$. Определим строку $S(G) \in \mathcal{B}^{n^2}$ как конкатенацию строк матрицы смежности $A(G)$: $\mathcal{L}(G) = [A[0,0], A[1,0], \ldots, A[n-1,0], A[0,1], \ldots, A[n-1,n-1]]$.

Каноническим представлением графа G называется граф $G' \sim G$, для которого $\mathcal{L}(G')$ лексикографически минимальна среди всех графов, изоморфных G: $G' = \underset{G_0 \sim G}{\arg\min}\, \mathcal{L}(G_0)$, где сравнение ведется лексикографически.

Исходя из определения, для всех графов из одного класса изоморфизма каноническое представление будет одним и тем же. Если бы была возможность по графу определять, что он является каноническим представлением своего класса, то предикат, допускающий только канонические представления был бы идеальным предикатом нарушения симметрии — он допускал бы ровно один граф из каждого класса изоморфизма.

Подход, в котором используется свойства канонического представления графа, был предложен в [2]. В этой работе исследуются свойства, которыми обладает любой канонический граф. Если при поиске графа на него дополнительно наложить свойства, характерные для канонического представления, то это может отбросить часть графов, при этом для каждого класса изоморфизма оставляя хотя бы один, потому что каноническое представление класса по условию удовлетворяет заданным свойствам. Чем строже будут такие свойства, тем больше графов будет отброшено во время поиска. В работе [2] были предложены два предиката нарушения симметрии, которые основаны на свойствах канонического представления графа.

Первый набор ограничений, линейный от количества вершин в графе, основывается на том, что строки матрицы смежности канонического представления графа упорядочены по возрастанию:

$$\mathsf{sb}_\ell = \bigwedge_{i=1}^{n-1} A[i] \preceq A[i+1],$$

где $\preceq$ означает «лексикографически меньше или равен».

Второй набор ограничений, квадратичный от количества вершин в графе, основан на более подробном исследовании свойств канонического представления графа и имеет следующий вид:

$$\mathsf{sb}^*_\ell = \bigwedge_{i<j} A[i] \preceq_{\{i,j\}} A[j],$$

где $\preceq_{\{i,j\}}$ означает лексикографическое сравнение последовательностей, у которых одновременно опущены элементы с индексами i и j.

1.3.2. Неизбежные подграфы

Другим подходом к нарушению симметрий в графах является использование неизбежных подграфов [5]. Для того, чтобы их ввести, понадобятся дополнительные определения.

Определение 18. Дополнением графа $G = \langle V, E\rangle$ называется граф $\overline{G} = \langle V, \overline{E}\rangle$, где $\overline{E} = (V \times V) \setminus \bigcup\limits_{v \in V} \langle v, v\rangle \setminus E$, то есть граф, в котором содержатся только ребра, которых не было в исходном графе ($(V \times V) \setminus \bigcup\limits_{v \in V} \langle v, v\rangle$ является множеством всех возможных ребер графа без учета петель).

Определение 19. Граф G_u называется неизбежным подграфом для графов размера не меньше n, если для любого графа из хотя бы n вершин либо он, либо его дополнение содержат подграф, изоморфный G_u: $\forall m \geqslant n\, \forall G \in \mathcal{G}_m \colon G_u \subset G \vee G_u \subset \overline{G}$.

Простейшим примером неизбежного подграфа является K_2 (граф из двух вершин и одного ребра), так как в любом графе либо есть ребро, либо есть две вершины, которые не соединены ребром.

Чуть более сложный пример: P_3 (граф из трех вершин, соединенных в «цепочку» двумя ребрами) является неизбежным подграфом для графов из трех и более вершин. На рисунке 1 изображены все графы из трех вершин (пунктиром обозначено отсутствие ребра) и каждый из них содержит либо два ребра подряд, либо два пунктирных ребра (ребра в дополнении графа) подряд. Заметим, что если какой-то граф G_u содержится в любом графе или его дополнении размера n, то и для всех бóльших графов он будет обладать таким свойством. Рассмотрим произвольный граф из $m > n$ вершин. Рассмотрим его подграф, состоящий из первых n вершин. Либо он, либо его дополнение содержит в себе G_u, а значит и весь граф размера m тоже обладает таким свойством. Поэтому, в частности, P_3 является неизбежным подграфом для всех графов из трех и более вершин.

В работе [5] рассматриваются методы нахождения наибольших (по числу ребер) неизбежных подграфов. На основе найденных графов предлагается следующий предикат нарушения симметрии. Рассмотрим

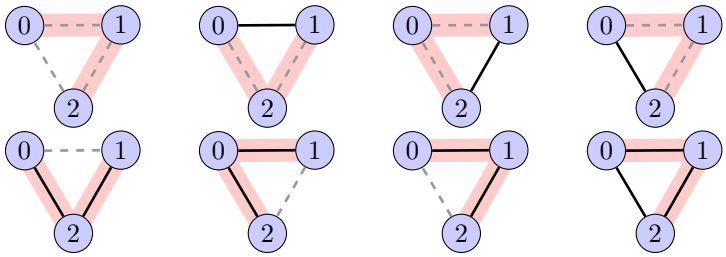

Рисунок 1 – Все возможные графы из 3 вершин. Пунктиром обозначено отсутствие ребра. Все графы содержат либо путь из двух ребер, либо путь из двух пунктирных ребер.

$G_u \in \mathcal{G}_n$ — неизбежный подграф для всех графов из m и более вершин. Пусть его вершины занумерованы от 0 до $n-1$. Пусть предикат $\text{USG-SBP}(G) = 1$ тогда и только тогда, когда в графе G присутствуют либо все ребра из графа G_u, либо ни одного:

$$\text{USG-SBP}(G) = e_1 \leftrightarrow e_2 \leftrightarrow \ldots \leftrightarrow e_{|E(G_u)|}, \tag{1}$$

где $e_1, \ldots, e_{|E(G_u)|}$ — элементы матрицы смежности, соответствующие ребрам из графа G_u.

Такой предикат является предикатом нарушения симметрии, потому что G_u содержится в любом графе или его дополнении размера не меньше m. Тогда граф, полученный из исходного перенумерацией вершин таким образом, чтобы подграф из первых n вершин (или его дополнение) совпадал с G_u, будет удовлетворять предикату φ и будет изоморфен исходному. Таким образом для каждого графа найдется изоморфный ему, удовлетворяющий предикату, а значит такой предикат является нарушением симметрии.

1.3.3. Канонизирующие множества

Возвращаясь к теме канонического представления графа, следует упомянуть другой подход его использования для устранения симметрий. Исходя из определения, каноническое представление графа единственно и для него есть простая формула: $can(G) = \underset{G_0 \sim G}{\arg\min}\, \mathcal{L}(G_0)$. Перепишем эту формулу с учетом определения изоморфизма (за S_n обозначим множество всех перестановок n элементов). Пусть $\min_\Pi(G) = \forall \pi \in \Pi \colon G \preceq \pi(G)$ — некоторый предикат, принимающий множество перестановок Π в качестве параметра. Тогда ка-

нонический представление G — такой граф $can(G)$, что выполнено $\min_{S_n}(can(G)) = \text{true}$. Предикат min_{S_n} допускает только один граф, потому что он допускает граф с минимальной (относительно некоторого линейного порядка) матрицей смежности, поэтому каноническое представление графа единственное.

Даже для маленьких n мощность множества всех перестановок очень велика: $|S_n| = n!$. Однако, в [6] показывается, что не обязательно перебирать все возможные перестановки.

Определение 20. Множество перестановок $\Pi \subset S_n$ называется канонизирующим, если $\forall G \in \mathcal{G}_n\colon\ \min_{\Pi}(G) \Leftrightarrow \min_{S_n}(G)$.

В работе [6] предлагается эффективный способ нахождения минимальных канонизирующих множеств для небольших графов. Для графов до 10 вершин получены впечатляющие результаты: вместо $10! = 3628800$ перестановок достаточно рассматривать только 7853 перестановки для проверки графа из 10 вершин на каноничность. Получающийся набор ограничений имеет простой вид:

$$\bigwedge_{\pi \in \Pi} G \prec \pi(G),$$

то есть прямая проверка предиката $min_{\Pi}(G)$ для заранее заданного Π.

1.3.4. BFS нумерация

Техники нарушения симметрии применяются ко многим задачам. Одной из таких задач является задача синтеза конечного автомата с заданными свойствами. В работе [4] рассматривается задача грамматического вывода конечного автомата и предикаты нарушения симметрии для синтеза автоматов в целом.

Основная идея нарушения симметрии, предложенная в этой работе, заключается в нумерации состояний автомата в порядке обхода в ширину, начиная из стартового состояния. Обход в ширину (BFS, breadth first search) — один из алгоритмов обхода вершин графа (или состояний автомата), который использует очередь — структуру данных с FIFO-порядком добавления и удаления элементов (first in, first out; первый пришел, первый ушел). Алгоритм работает следующим образом. Изначально в очередь вершин кладется первая вершина. Далее, пока очередь

не пуста, из очереди берется очередная вершина. Для этой вершины находятся все ребра, которые исходят из нее в еще не рассмотренную вершину, и все такие вершины добавляются в очередь. Порядок обхода вершин в таком процессе определяется тем, какой по счету вершина была добавлена в очередь. Псевдокод обхода графа в ширину приведен в листинге 1.

Листинг 1 – Псевдокод обхода BFS

```
void BFS(Graph G) {
    Queue<Node> queue = new Queue<>();
    Set<Node> used = new HashSet<>();
    queue.put(0);
    while (!queue.isEmpty()) {
        Node v = queue.poll();
        for (Node u: v.children) {
            if (u ∉ used) {
                queue.add(u);
                used.add(u);
            }
        }
    }
}
```

Такой метод нарушения симметрии для каждого класса изоморфизма автоматов оставляет ровно один, таким образом являясь идеальным нарушением симметрии для задачи синтеза автоматов.

Однако, несмотря на то, что подобный метод можно применить и к графам, он будет гораздо менее эффективен из-за наличия в графе большего количества симметрий. В отличие от автомата, в графе нет выделенного стартового состояния, а также порядка на ребрах, который в автомате обеспечивается метками на ребрах. Данная работа уменьшает это отличие, предлагая набор методов, которые частично ограничивают эти симметрии в графах.

1.4. Средства решения построенных моделей

Самыми известными задачами выполнимости являются задачи SAT, CSP и SMT (выполнимость булевых формул, выполнимость ограничений и выполнимость формул в теориях). Существует множество программных средств для решения этих задач и с каждым готом появляются

новые. Однако многие из них используют идеи одного из двух алгоритмов — DPLL и CDCL.

1.4.1. Алгоритм DPLL

Алгоритм Дэвиса-Патнема-Логемана-Лавленда (DPLL) представляет собой алгоритм полного поиска с возвратом решения для булевой формулы. Поэтому изначально он является алгоритмом решения задачи SAT. Однако, его можно обобщить и на остальные задачи выполнимости. Алогритм заключается в рекурсивном обходе дерева поиска и правилах упрощения формулы. Стоит заметить, что алгоритм оперирует конъюнктивным нормальным представлением булевой формулы.

Если дизъюнкта содержит одну переменную или ее отрицание, то эта переменная не может иметь ложное значение, иначе вся дизъюнкта, а, следовательно, и вся формула будут ложны. Поэтому все такие дизъюнкты можно убрать из формулы, а полученные значения переменных подставить в формулы, таким образом упростив ее. Поэтому первый этап алгоритма заключается в последовательном упрощении формулы.

Если какая-то переменная входит в формулу либо всегда без отрицания, либо только с отрицанием, то назначив ей истинное или, соответственно, ложное значение, можно сделать истинными все дизъюнкты, в которых она содержится. Поэтому второй этап алгоритма заключается в поиске всех таких переменных и последующем упрощении формулы.

После всех этапов упрощения формулы, когда описанные правила уже не могут упростить полученную формулу, начинается этап разбиения на две подзадачи. Выбирается произвольная переменная и перебираются ее возможные значения — истина или ложь. Каждое значение порождает подзадачу, в которой эта переменная имеет конкретное значение, и эта подзадача рекурсивно решается этим же алгоритмом. Если в результате упрощений формулы не останется ни одной дизъюнкты, то формула является удовлетворимой. Иначе, если в процессе упрощения одна из дизъюнкт станет пустой, это означает конфликт (при данных значениях переменных формула всегда будет ложной) и требуется откатываться к предыдущим решениям. В листинге 2 приведен псевдокод алгоритма DPLL.

Листинг 2 – Псевдокод алгоритма DPLL

```
1  boolean DPLL(List<Clause> ϕ) {
2      if (ϕ.isEmpty())
3          return true;
4      if (ϕ contains empty clause)
5          return false;
6      ϕ = simplify(ϕ);
7      x = pick-variable(ϕ);
8      return DPLL(ϕ ∧ [l = true]) ||
9             DPLL(ϕ ∧ [l = false]);
10 }
```

1.4.2. Алгоритм CDCL

Алгоритм обучения дизъюнкт на основе конфликтов (conflict-driven clause learning, CDCL) является усовершенствованием алгоритма DPLL. Основными отличиями от DPLL являются обучение новым дизъюнктам и нехронологический возврат.

Общая схема поиска решения остается такой же, как и в DPLL. Для эффективного упрощения формулы поддерживается граф импликаций — ориентированный граф, вершинами которого являются выражения $x_k = 0$ или $x_k = 1$ (где x_k — одна из переменных в формуле), а ребра устроены следующим образом: если из вершин $x_{i_1} = b_1, \ldots, x_{i_n} = b_n$ (и только из них) ведут ребра вершину $x_{i_0} = b_0$, то из исходной булевой формулы выводится логическое следствие $(x_{i_1} = b_1) \wedge \ldots \wedge (x_{i_n} = b_n) \to x_{i_0} = b_0$, где $b_i \in \mathcal{B}$. Благодаря такому графу при каждой новой открытой переменной можно быстро упрощать формулу — из графа удаляется вершина, которая соответствует значению новой открытой переменной. После этого в графе находятся все вершины, у которых нет входящих ребер — все такие вершины применяются к значениям переменных (если в вершину $x_k = b$ нет входящих ребер, то переменная x_k не может иметь значение, отличное от b).

Отличия от алгоритма DPLL появляются при нахождении конфликта — если из графа импликаций были выведены вершины $x_k = 0$ и $x_k = 1$ для некоторой переменной x_k. В таком случае строится разрез (разбиение вершина графа на два подмножества), в одной части которого находятся две конфликтные вершины ($x_k = 0$ и $x_k = 1$), а в другой — все остальные. Далее находятся все ребра, пересекающие разрез

(два конца ребра находятся в разных частях разреза). Составляется список стартовых вершин всех таких ребер ($x_{i_1} = b_1, \ldots, x_{i_n} = b_n$). Можно заметить, что все вершины из данного списка не могут быть выполнены, так как их одновременное выполнение приводит к конфликту. Таким образом, к исходной формуле можно добавить новую дизъюнкту $(\neg(x_{i_1} = b_1) \vee \ldots \vee \neg(x_{i_n} = b_n))$. После модификации исходной формулы необходимо вернуться к самому раннему узлу, в котором выбиралось значение для одной из переменных новой дизъюнкты, потому что уже пройденный путь не является корректным из-за изменения формулы. Этот шаг называется нехронологическим возвратом.

Данный алгоритм, как и DPLL, обобщается на другие задачи выполнимости и широко распространен в программах для решения задач SAT и CSP. Псевдокод алгоритма CDCL приведен в листинге 3.

Листинг 3 – Псевдокод алгоритма CDCL

```
boolean CDCL(List<Clause> ϕ) {
    Graph implicationGraph = new Graph();
    while (not end) {
        x = pick-variable(ϕ);
        assign-new-value(x); // next unseen value
        ϕ = simplify(ϕ, implicationGraph);
        if (ϕ.isEmpty())
            return true;
        if (conflict) {
            List<(Variable, boolean)> conflictList =
                getConflictList(implicationGraph);
            ϕ += {(x, !value) for (x, value) ∈ conflictList};
            backtrack-to(earliest state from conflictList);
        }
    }
    return false;
}
```

1.4.3. Средства решения задачи SAT

Программные средства для решения задачи SAT называются SAT-решателями (SAT-solvers). Большинство таких решателей принимают на вход файл, содержащий конъюнктивную нормальную формулу в формате DIMACS, на выходе они дают решение задачи в том же формате. Формат DIMACS стал стандартом для многих соревнований, а, следовательно, и SAT-решателей. Решатели делятся на однопоточные и много-

поточные. Среди однопоточных можно перечислить следующие известные программы: lingeling [7], glucose [8], cryptominisat [9]. Среди многопоточных: plingeling и treengeling [7], glucose-syrup [8].

1.4.4. Средства решения задачи CSP

Задача CSP (Constraint Satifsaction Problem) заключается в поиске значений набора переменных, которые удовлетворяют определенным ограничениям. Совокупность переменных и ограничений над ними называется моделью задачи CSP. Каждая переменная имеет множество возможных значений. Большинство систем для решения задач CSP поддерживают булевы переменные и целочисленные переменные из конечного промежутка значений. Пример такой модели на языке MiniZinc представлен в листинге 4. В нем переменная a имеет булев тип, а переменные b и c являются целыми числами из диапазона $[1, 10]$. Также там присутствуют два ограничения, которые должны выполняться для этого набора переменных. Для этой модели правильными ответами, среди прочих, являются $[a = \text{true}, b = 4, c = 5]$ и $[a = \text{false}, b = 6, c = 1]$.

Листинг 4 – Пример CSP модели на языке MiniZinc

```
1 var bool: a;
2 var 1..10: b;
3 var 1..10: c;
4 constraint b * b + c * c < 100;
5 constraint a <-> (b < c);
6 solve satisfy;
```

Язык MiniZinc является языком описания CSP-моделей достаточно высокого уровня. Язык BEE является языком более низкого уровня, поэтому для описания предыдущей модели необходимо вводить новые переменные для промежуточных выражений. Одно из возможных описаний предыдущей модели на языке BEE приведено в листинге 5. Решению $[a = \text{true}, b = 4, c = 5]$ первой модели соответствует следующее решение второй модели: $[a = \text{true}, b = 4, c = 5, b\_sq = 16, c\_sq = 25, bc\_sum = 41]$.

Принимая во внимание сказанное выше, можно выделить следующие наиболее популярные средства для решения задачи CSP:

— Choco 3 [10]

Листинг 5 – Пример CSP модели на языке BEE

```
new_bool(a)
new_int(b, 1, 10)
new_int(c, 1, 10)
new_int(b_sq, 1, 100)
new_int(c_sq, 1, 100)
new_int(bc_sum, 1, 200)
int_times(b, b, b_sq)
int_times(c, c, c_sq)
int_plus(b_sq, c_sq, bc_sum)
int_lt(bc_sum, 100)
int_lt_reif(b, c, a)
solve satisfy
```

CSP-решатель, написанный полностью на Java. Его отличительной особенностью является наличие модуля choco-graph, который добавляет новые типы переменных: неориентированный и ориентированный граф.

— MiniZinc [11]
Одновременно язык задания CSP-моделей и CSP-решатель. Он позволяет задавать сложные модели с помощью простого, но мощного языка. Кроме самого MiniZinc существует множество CSP-решателей, которые принимают на вход CSP-модель на языке MiniZinc, например GECODE и Opturion CPX.

— BEE [12] (Ben-Gurion-University Equi-propagation Encoder)
Транслятор CSP-моделей в SAT, а также низкоуровневый языком задания CSP-моделей. Он использует несколько настраиваемых видов кодирования целых чисел в набор булевых переменных, а также применяет множество оптимизаций к получаемой формуле на этапе трансляции.

1.5. Актуальные задачи и области применимости

Графы являются универсальным инструментом, который используется во множестве различных областей разработки и исследований. Они часто возникают в качестве модели, которая описывает различные задачи. Именно благодаря широкому распространению теория графов активно изучается и каждый год открываются новые факты и разрабатываются продвинутые алгоритмы по работе с графами. В данном раз-

деле приводятся несколько задач поиска графов, в которых актуально применение методов нарушения симметрии, описанных в данной работе.

1.5.1. Задача Турана

Одной из известных задач в комбинаторике и поиске графов является задача Турана: какое максимальное число ребер может иметь граф из n вершин, который не содержит в себе подграфа, изоморфного K_r? Впоследствии была сформулирована более общая задача, где вместо полного графа размера r рассматривается набор произвольных графов $G_1, G_2, \ldots, G_k$.

Определение 21. $ex(n; G_1, \ldots, G_k)$ — максимальное число ребер в графе из n вершин, который не содержит подграфа, изоморфного одному из графов $G_1, \ldots, G_k$.

$EX(n; G_1, \ldots, G_k)$ — множество графов из n вершин (называемых экстремальными), которые не содержат подграфа, изоморфного одному из графов $G_1, \ldots, G_k$ и имеют максимальное число ребер, то есть $ex(n; G_1, \ldots, G_k) = |EX(n; G_1, \ldots, G_k)|$.

Точное решение задачи поиска значения $ex(n; G_1, \ldots, G_k)$ известно для очень маленького количества наборов $G_1, \ldots, G_k$. Однако, известно множество асимптотических оценок и границ для решения этой задачи.

Для набора графов $G_1, \ldots, G_k$, среди которых нет ни одного двудольного, известна общая асимптотическая формула:

$$ex(n; G_1, \ldots, G_k) \sim \left(1 - \frac{1}{\chi_{min} - 1}\right)\binom{n}{2}, \qquad n \to \infty, \tag{2}$$

где $\chi_{min} = \min\{\chi(G_i) \mid 1 \leqslant i \leqslant k\} \geqslant 3$ — минимальное хроматическое число среди всех графов $G_1, \ldots, G_k$ (так как среди графов нет двудольных, то все хроматические числа, как и их минимум, будут больше или равны трем). В случае, если среди графов есть двудольные, не известно общей формулы, похожей на (2).

Известным результатом является $ex(n; C_3) = \lfloor n^2/4 \rfloor$, который достигается на полном двудольном графе $K_{\lfloor n/2 \rfloor, \lceil n/2 \rceil}$. Точное значение $ex(n; C_4)$ известно для чисел специального вида: $n = q^2 + q + 1$, где $q = 2^k$ [13] или $q = p^k$ для простых $p > 13$ [14]. Также

известные следующие асимптотические оценки: $ex(n; C_4) \sim \frac{1}{2}n^{3/2}$ и $ex(n; C_3, C_4) \sim \frac{1}{2\sqrt{2}}n^{3/2}$ [15].

Не смотря на то, что задача не кажется важной на первый взгляд, она находит неожиданные применения в разных областях, например [16]. Она естественным образом формулируется как задача поиска графов и многие результаты были получены сведением с задачам CSP и SAT с использованием методов нарушения симметрий, например [2].

1.5.2. Раскраска графа и теорема Рамсея

Задача раскраски графа заключается в назначении каждому ребру или вершине графа одного из k цветов так, чтобы выполнялись некоторые свойства. Самыми известными являются задачи вершинной и реберной раскраски со следующей формулировкой.

Определение 22. Реберная раскраска графа G — функция $f\colon E \to \{1..k\}$, такая, что для любых смежных ребер $e_1, e_2 \in E(G)$ они имеют различные цвета: $f(e_1) \neq f(e_2)$.

Определение 23. Вершинная раскраска графа G — функция $f\colon V \to \{1..k\}$, такая, что для любых двух вершин $u, v \in V(G)$, соединенных ребром ($\langle u, v\rangle \in E(G)$), они имеют различные цвета: $f(u) \neq f(v)$.

Эти задачи являются важными и часто встречаются на практике. Задача реберной раскраски возникает в теории расписаний и даже в распределении частот в оптоволоконной сети [17].

Одним из самых известных применений задачи вершинной раскраски является задача распределения регистров при компиляции программ [18]. Для оптимизации времени работы скомпилированной программы необходимо, чтобы как можно больше переменных хранилось в регистрах, чтобы свести обращение к оперативной памяти к минимуму. Для каждой переменной определяется время жизни — промежуток инструкций, в котором значение этой переменной должно быть доступным. Для того, чтобы упростить алгоритм распределения регистров, программа приводится к SSA-форме, в которой каждой переменной присваивается значение только один раз, таким образом время жизни каждой переменной представляется непрерывным отрезком инструкций. Задача распределения регистров заключается в присвоении

каждой переменной регистра таким образом, чтобы никакие две переменные с пересекающимися временами жизни не хранились в одном и том же регистре.

Для сведения этой задачи к вершинной раскраске графа используется следующая модель: переменные являются вершинами графа, а ребра соединяют такие пары вершин, что время жизни соответствующих им переменных пересекается. Множество цветов представляет собой множество доступных регистров. Если существует вершинная раскраска данного графа в такое множество цветов, то она соответствует оптимальному распределению регистров исходной задачи. В противном случае существует набор методов и эвристик, с помощью которых одна из переменных выгружается в память, и в новом графе снова ищется вершинная раскраска.

На практике, для решения этой задачи используются приближенные и жадные методы, например [19]. Они работают за полиномиальное время и существенно превосходят все точные методы по скорости, сохраняя при этом относительно хорошее решение благодаря множеству эвристик. Однако, в ряде ситуаций, время работы программы очень важно, а время компиляции отходит на второй план. В таких ситуациях и находят применение точные методы раскраски графа, для которых и применимы предлагаемые в работе методы нарушения симметрии.

Другой задачей, связанной с графовой раскраской и поиском графов, является нахождение чисел Рамсея.

Утверждение 1 (Теорема Рамсея). Для любых чисел $a_1, a_2, \ldots, a_k$ существует число $R(a_1, \ldots, a_k)$ такое, что для любой раскраски ребер полного графа K_R в k цветов существует либо полный подграф из a_1 вершин со всеми ребрами 1-ого цвета, ... либо полный подграф из a_k вершин со всеми ребрами k-ого цвета.

Числа $R(a_1, \ldots, a_k)$ называются числами Рамсея и их нахождение является сложной и открытой задачей. Только несколько чисел Рамсея известно на данный момент. Данная задача является настолько сложной из-за огромного количества графов и графовых раскрасок, и методы нарушения симметрии могут существенно ускорить процесс их перебо-

ра. Такие техники все чаще применяются в последнее время, например в [20], [21] и [22].

Выводы по главе 1

Существует множество актуальных задач поиска графов, в которых остро встает проблема наличия большого количества изоморфмных решений. Существуют различные техники решения таких задач, которые применяются на практике. Для ускорения поиска решений существует множество разнообразных методов нарушения симметрий, обладающих разными свойствами и лучше подходящих для различных видов задач. Однако не существует ни одного метода, который был бы применим в широкой области графовых задач и сочетал в себе простоту и эффективность реализации. Предикаты, которые устраняют все симметрии существуют только для малых графовых задач, а остальные методы демонстрируют разное ускорение в зависимости от конкретной задачи. Поэтому актуальной задачей является поиск новых методов нарушения симметрии, которые превосходят существующие в применимости или ускорению поиска.

ГЛАВА 2. ПРЕДИКАТЫ НАРУШЕНИЯ СИММЕТРИИ

В данной главе приводятся основные теоретические результаты, предлагаемые в работе. Она представляет собой подробное описание предикатов нарушения симметрии, а также доказательства их корректности и описания построения CSP моделей.

2.1. Представление графа в CSP модели

Самым распространенным способом представить граф $G \in \mathcal{G}_n$ в CSP модели является введение переменных, задающих матрицу смежности. В зависимости от модели и используемой программы для её решения матрицу можно задавать целиком (n^2 булевых переменных) или только верхним треугольником ($n(n-1)/2$ булевых переменных). Например, при использовании BEE между подходами нет разницы, так как при трансляции модели в SAT транслятор BEE сам заменит равные элементы матрицы на одну переменную (при указании ограничений, рассматриваемых далее).

Дополнительно на эти переменные накладываются ограничения, гарантирующие корректность матрицы смежности, такие как симметричность (3) и отсутствие петель (4):

$$\forall u, v\colon A[u, v] = A[v, u], \tag{3}$$

$$\forall v\colon A[v, v] = \text{false}. \tag{4}$$

Матрица смежности представляет собой полное описание графа, поэтому для наложения любых ограничений на граф, которые возникают в задаче, достаточно введенных переменных.

2.2. Нарушение симметрий на основе обхода BFS

В работе [4] был предложен метод нарушения симметрий в конечных автоматах, основанный на упорядочивании состояний в порядке BFS. В данной работе предлагается применение аналогичного подхода к графам.

Определение 24. Предикат $\varphi_{\text{BFS}}(G) = \text{true}$ тогда и только тогда, когда граф G занумерован в порядке обхода BFS, то есть существует такой

обход BFS, что он обходит все вершины в том же порядке, в каком им соответствуют метки $(V[0], V[1], V[2], \ldots, V[n-1])$.

Теорема 1. Предикат φ_{BFS} является корректным предикатом нарушения симметрии для множества связных графов.

Доказательство. Рассмотрим произвольный связный граф и какой-то его обход BFS (всегда существует для связных графов). Расставим метки на вершинах в порядке обхода BFS. Новый граф изоморфен исходному (так как является его перенумерацией) и удовлетворяет φ_{BFS} по определению. Значит для любого связного графа найдется изоморфный ему, который удовлетворяет φ_{BFS}. □

Одним их преимуществ такого предиката на разреженных графах является то, что в лучшем случае достаточно одного ребра, чтобы запретить граф целиком. Пример приведен на рисунке 2. Из пяти изоморфных представлений графа S_4 только два графа имеют подходящую нумерацию BFS. Для них красными стрелками нарисовано подходящее дерево BFS. Для остальных графов из этого класса подходящего обхода BFS не существует. Это легко понять, так как если вершина 0 имеет только одну смежную вершину, то такая вершина обязана иметь номер 1. Такие «запрещающие» ребра показаны на рисунке красным волнистым ребром. Благодаря им средство, решающее задачу SAT, при поиске может существенно заранее отбрасывать ветви пространства поиска, которые соответствуют этим графам, имея информацию лишь о небольшом количестве ребер.

Для того, чтобы закодировать φ_{BFS} в CSP требуется ввести дополнительные переменные. Для каждой вершины v кроме нулевой (корня BFS-дерева) введем переменную p_v, которая будет хранить номер родителя вершины v в дереве BFS. Для того, чтобы эти переменные задавали корректное дерево BFS, накладывается ограничение, которое следует из определения обхода BFS:

$$\forall u, v\colon p_v = u \Leftrightarrow (A[u, v] \wedge \nexists\, k < u\colon A[k, v])\,. \tag{5}$$

В ограничении (5) представлено следующее: вершина v была добавлена в очередь при просмотре вершины u только в том случае, если есть ребро (u, v), и в вершину v не ведет ребро ни из какой вершины с но-

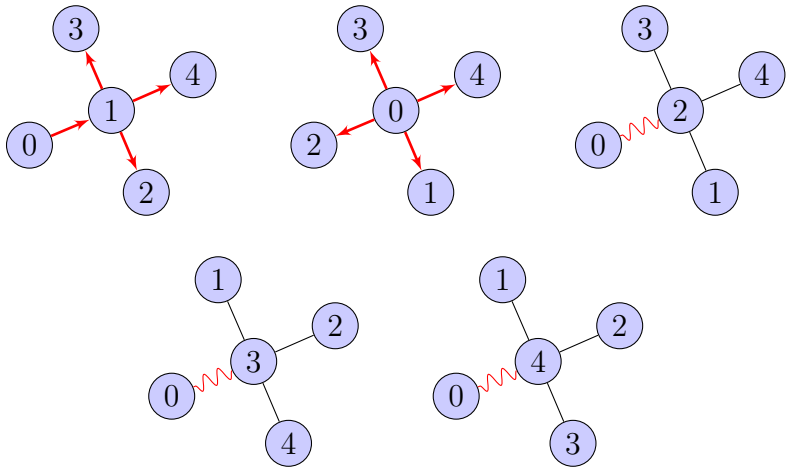

Рисунок 2 – Пример класса изоморфизма графа из 5 вершин, из которых только два занумерованы в порядке обхода в ширину. Дерево обхода в ширину изображено красными стрелками в графе. Волнистые ребра обозначают «запрещающие» ребра, наличие которых позволяет отбросить граф целиком.

мером, меньшим u. Таким образом, это ограничение задает корректный обход BFS по определению.

Для повышения эффективности в модель добавляются свойства, которые следуют из обхода BFS:

$$\forall v\colon p_v < v, \tag{6}$$

$$\forall v\colon p_v \leqslant p_{v+1}. \tag{7}$$

Ограничение (6) получается следующим образом: для каждой новой вершины из очереди рассматриваются только те, которые еще не были добавлены в очередь. Так как изначально в очереди лежит только вершина с номером 0, в каждый момент времени все вершины, которые были или есть в очереди, имеют меньшие метки, чем те вершины, которые еще не были рассмотрены. Отсюда следует, что метка родителя всегда меньше, чем метка ребенка в дереве BFS.

Аналогично можно заметить, что так как вершины обходятся по возрастанию меток, если $u < v$, то все дети v будут иметь большие метки, чем дети u. Отсюда следует, что если $u < v$, то метка родителя u меньше или равна, чем метка родителя v, следовательно ограничение (7) выполняется.

2.3. Ограничение стартовой вершины

Рассмотренный подход отсеивает часть изоморфных графов, однако он никак не фиксирует стартовую вершину. Из-за этого предикат допускает все изоморфные графы, которые отличаются лишь выбором стартовой вершины. Пример таких графов приведен на рисунке 3.

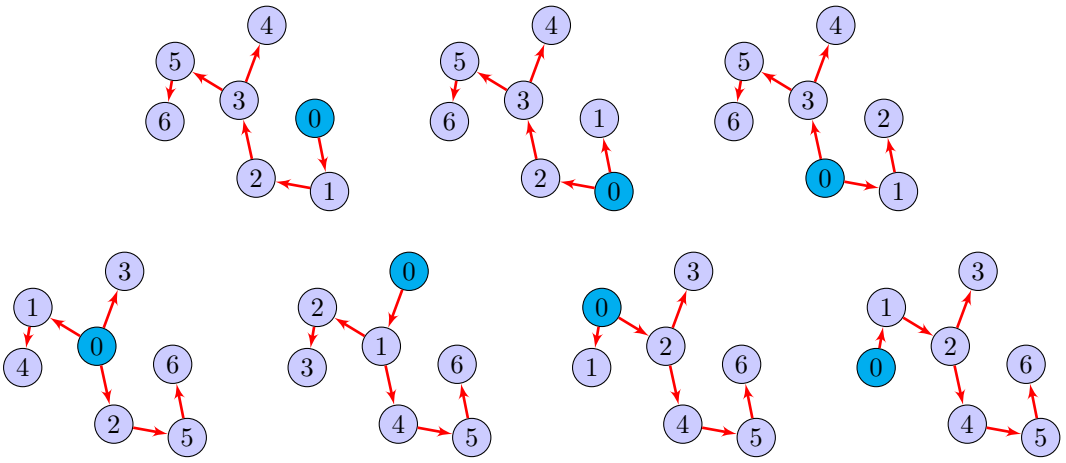

Рисунок 3 – Пример семи изоморфных графов, которые допускаются предикатом φ_{BFS}. Голубым цветом выделена стартовая вершина, пробегающая все возможные вершины в графе.

В качестве способа ограничить множество возможных стартовых вершин предлагается потребовать, чтобы стартовая вершина имела максимальную степень в графе.

Определение 25. Предикат $\varphi^{\deg}_{\text{BFS}}(G) = 1$ тогда и только тогда, когда $\varphi_{\text{BFS}}(G) = 1$ и вершина с номером 0 имеет максимальную степень в графе: $\deg v_0 = \max\limits_{v \in V(G)} \deg v$.

Теорема 2. Предикат $\varphi^{\deg}_{\text{BFS}}$ является корректным предикатом нарушения симметрии для множества связных графов.

Доказательство. Доказательство аналогично доказательству теоремы 1, только вместо произвольной вершины необходимо выбрать вершину с максимальной степенью. □

Новый предикат фильтрует еще больше изоморфных решений, хоть и не все. В графе может быть несколько вершин с максимальной степенью и новое ограничение не поможет разделить такие случаи. Из семи графов на рисунке 3 только один допускается предикатом $\varphi^{\deg}_{\text{BFS}}$, потому что только одна вершина в графе на рисунке имеет степень 3, все остальные вершины имеют меньшую степень.

Для представления $\varphi^{\deg}_{\text{BFS}}$ в CSP-модели необходимо ввести переменные, которые соответствуют весам вершин в графе (deg_i) и ограничения, которые задают их корректность и накладывают условие на стартовую вершину:

$$\forall v\colon \deg_v = \sum_{u \in V} A[u, v], \tag{8}$$

$$\max\_\deg = \max_{v \in V} \deg_v, \tag{9}$$

$$\deg_0 = \max\_\deg. \tag{10}$$

Формула (8) определяет степень каждой вершины, (9) — максимальную степень вершины в графе. Непосредственно условие максимальной степени стартовой вершины, которое вводится в новом предикате, задается ограничением (10).

2.4. Уточнение структуры BFS-дерева

Предикат $\varphi^{\deg}_{\text{BFS}}$ требует, чтобы граф был занумерован в порядке некоторого BFS-обхода и накладывает ограничения на стартовую вершину. Однако, для любой фиксированной вершины порядок нумерации ее детей (в дереве BFS) не определен. Таким образом даже для зафиксированной стартовой вершины может существовать несколько различных обходов BFS. На рисунке 4 изображены изоморфные графы, удовлетворяющие $\varphi^{\deg}_{\text{BFS}}$. У графа на рисунке зафиксирована стартовая вершина как вершина с максимальной степенью. Однако из этой вершины выходит три разные «ветви» графа — пути длины 1, 2 и 3. При обходе BFS можно выбрать любой из $3! = 6$ порядков обхода этих ветвей, что и порождает шесть изоморфных графов, удовлетворяющих $\varphi^{\deg}_{\text{BFS}}$. Для того, чтобы разделять такие графы, необходимо каким-то образом ограничить структуру дерева BFS.

Определение 26. Рассмотрим некоторое ориентированное дерево T. Весом поддерева с корнем в вершине u (далее — поддерева u) назовем количество вершин в этом поддереве (включая корень): $w(u) = 1 + \sum\limits_{v\colon (u,v) \in E(T)} w(v)$.

Для корня дерева его вес поддерева равен количеству вершин во всем дереве, а для листа вес поддерева равен единице. Пример дерева

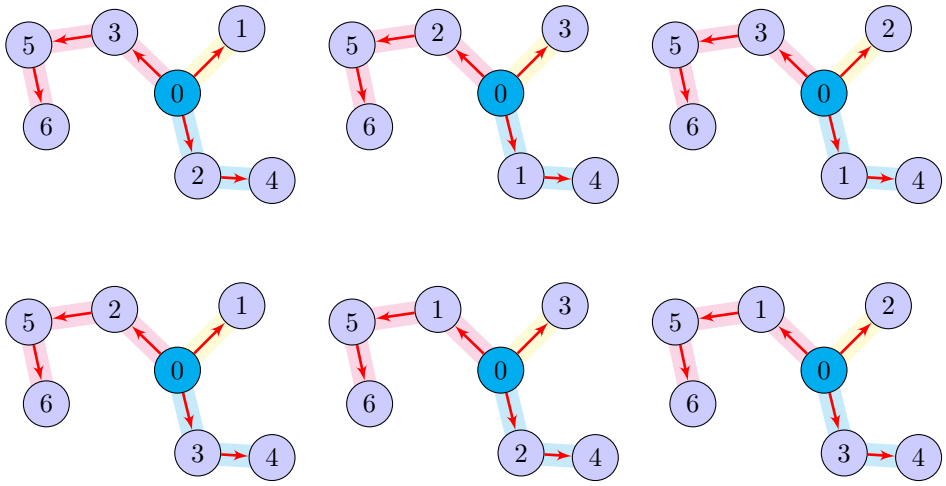

Рисунок 4 – Шесть изоморфных графов, удовлетворяющих $\varphi^{\deg}_{\text{BFS}}$. Цветом выделены разные «ветви» графа.

с обозначенными весами поддеревьев (для детей стартовой вершины) приведен на рисунке 5.

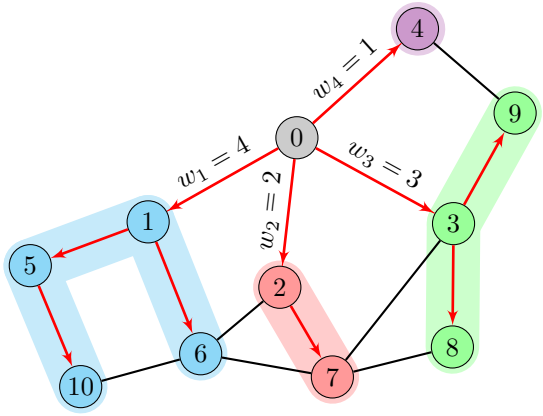

Рисунок 5 – Пример графа с указанными весами поддерева детей корня. Цветом выделены вершины из каждого поддерева.

Используя эту информацию о структуре дерева на него можно наложить дополнительные ограничения.

Определение 27. Предикат $\varphi^{\deg+w}_{\text{BFS}}(G) = 1$ тогда и только тогда, когда $\varphi^{\deg}_{\text{BFS}}(G) = 1$ и для каждой вершины в дереве BFS (которое соответствует нумерации) последовательность весов поддеревьев детей этой вершины упорядочена по невозрастанию: $\forall\, v \in V(G) \colon \forall\, t, u < t \colon\ (p_u = v \wedge p_t = v) \Rightarrow w(u) \geqslant w(t)$.

Теорема 3. Предикат $\varphi^{\deg+w}_{\text{BFS}}$ является корректным предикатом нарушения симметрий для множества связных графов.

Для доказательства теоремы понадобится доказательство вспомогательных утверждений.

Лемма 1. Пусть $G \in \mathcal{G}_n$ и выполнено $\varphi_{\text{BFS}}(G) = 1$. Пусть $\deg v_0 = m$ и w_i — вес поддерева i-ого ребенка вершины v_0. Пусть $w_i \leqslant w_{i+1}$ для некоторого i. Пусть G' — граф, полученный из G перестановкой i-ого и $i+1$-ого детей вершины v_0. Пусть $\{w'_1 \dots w'_m\}$ — веса поддеревьев вершин в новом графе. Тогда $\forall k \neq i, i+1 \colon w_k = w'_k$, $w'_i \geqslant w'_{i+1}$.

Доказательство. Пусть W_k — все вершины в поддереве k-ого ребенка вершины v_0 в дереве BFS (W'_k — аналогично в графе G'). Пусть $W_{k,j}$ — пересечение W_k с множеством вершин на j-ом слое в дереве BFS (находящиеся от v_0 на расстоянии j ребер). Заметим, что $w_k = |W_k|$, $W_k = \bigcup_{j=1}^{n} W_{k,j}$. Замечание: здесь и далее под равенством вершин из разных графов G и G' подразумевается равенство с учетом перестановки, то есть $\forall k \colon v_k = v'_k$.

Для доказательства рассмотрим следующие утверждения:

Утверждение 1. $\forall\, k \neq i, i+1\ \forall j \colon W_{k,j} = W'_{k,j}$.

Докажем по индукции:

База индукции: $j = 1, W_{k,1} = \{v_k\}, W'_{k,1} = \{v'_k\}$, значит, $W_{k,1} = W'_{k,1}$.

Переход: Пусть $W_{k,j} = W'_{k,j}$. Следующий слой формируют концы ребер, которые исходят из текущего слоя и не достижимы ни из каких вершин с меньшим номером:

$$W_{k,j+1} = \{v \mid \langle u, v\rangle \in E(G), u \in W_{k,j}, \nexists\, t < u \colon \langle t, v\rangle \in E(G)\}. \qquad (11)$$

Менялись местами только вершины i и $i+1$, которые либо обе больше, либо обе меньше, чем k. Следовательно, множество ребер $\langle t, v\rangle, t < u$ не изменилось. В силу индукционного предположения отсюда следует, что и множества вершин следующего слоя будут равны: $W_{k,j+1} = W'_{k,j+1}$.

Утверждение 2. $\forall j \colon W_{i,j} \subseteq W'_{i,j}$.

База индукции: $j = 1, W_{i,1} = \{v_i\} = \{v'_i\} = W'_{i,1}$.

Переход: Пусть $W_{i,j} \subseteq W'_{i,j}$. Множество для следующего слоя строится по формуле 11. Заметим, что множество $W'_{i,j+1}$ могло только уменьшиться по сравнению с $W_{i,j+1}$, потому что новая вершина v'_{i+1} находится левее v'_i, поэтому квантор несуществования пробегает больший набор значе-

ний, следовательно количество удовлетворяющих вершин могло только уменьшиться. Следовательно, $W_{i,j+1} \subseteq W'_{i,j+1}$.

Утверждение 3. $\forall j : W_{i+1,j} \supseteq W'_{i+1,j}$.

Доказательство полностью аналогично доказательству утверждения 2, только вместо добавления новой вершины в квантор несуществования одна вершина оттуда убирается, таким образом только увеличивая результирующее множество удовлетворяющих вершин.

Из утверждений 2 и 3 следует требуемое неравенство: $w'_i \geqslant w_{i+1} \geqslant w_i \geqslant w'_{i+1}$, следовательно $w'_i \geqslant w'_{i+1}$. Из утверждения 1 следует, что остальные вершины не изменили вес своего поддерева. Следовательно, условие теоремы выполняется. □

Пример такой перенумерации и последующего изменения весов представлен на рисунке 6.

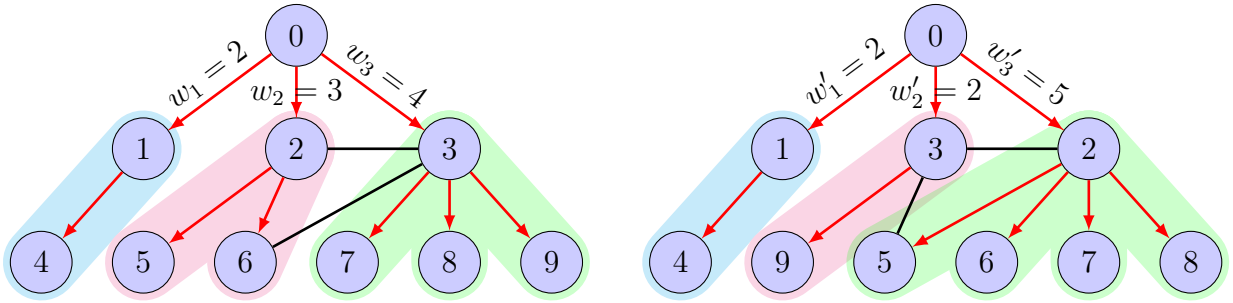

Рисунок 6 – Дерево BFS до и после перенумерации вершин.

Лемма 2. Пусть $G \in \mathcal{G}_n$, $\varphi_{\text{BFS}}(G) = 1$ и $\deg v_0 = k$. Тогда существует граф $G' \sim G$, $\deg v'_0 = k$ такой, что $w'_1 \geqslant w'_2 \geqslant \ldots \geqslant w'_k$, где w'_i — вес поддерева i-ого ребенка вершины v'_0.

Доказательство. Пусть $[w_1, \ldots, w_k]$ — последовательность весов поддеревьев детей вершины v_0. Пусть в ней есть хотя бы одна инверсия (инверсия относительно убывающего порядка, то есть пара $i < j$ такая, что $w_i < w_j$). Тогда существует и пара соседних элементов, которая является инверсией. Из леммы 1 следует, что если поменять такие две вершины местами, то веса могут измениться, но они перестанут быть инверсией (после перестановки будет выполнено $w_i \geqslant w_{i+1}$). Таким образом, пока в последовательности весов графа есть инверсии, по лемме 1 можно построить новый граф, в котором инверсий строго меньше. Следовательно, так как изначально инверсий могло быть только конечное

количество, а меньше нуля их быть не может, такая последовательность графов $G_1, G_2, \ldots$ в итоге приведет к графу G', у которого веса поддеревьев детей вершины v'_0 будут отсортированы в невозрастающем порядке. □

Доказательство (теоремы 3). Из леммы 2 следует, что можно построить граф, в котором отсортирована последовательность весов поддеревьев детей корня. Однако из деталей доказательства лемм 1 и 2 следует, что такое преобразование графа затрагивает только вершины из поддерева в дереве BFS. Таким образом если воспользоваться леммой 2 для произвольной вершины в графе, то структура дерева BFS для вершин из более верхних слоев и соседних поддеревьев не изменится.

По лемме 2 существует граф G_1 с отсортированной последовательностью весов корня. Применяя лемму 2 для каждого ребенка корня можно получить граф G_2, у которого отсортированы веса корня и всех его детей. Продолжая эту процедуру слой за слоем, можно получить граф G', у которого для каждой вершины будет выполнено свойство отсортированности весов. □

Таким образом, всегда существует граф, в котором веса поддеревьев отсортированы в невозрастающем порядке для каждой вершины. Следует заметить, что порядок, в котором отсортированы дети каждой вершины, важен. Предикат, допускающий графы с весами детей, упорядоченными в порядке неубывания, не является предикатом нарушения симметрии. Простейший контрпример, граф, никакая нумерация которого не удовлетворяет такому предикату, приведен на рисунке 7. Это происходит из-за того, что при формировании дерева BFS вершины обходятся в порядке очереди, и более ранние вершины имеют больше возможностей добавить вершины в очередь, чем более поздние вершины. Поэтому деревья BFS в некоторой степени несимметричны, они имеют тенденцию иметь более «тяжелые» левые поддеревья, чем правые.

Среди графов на рисунке 4 только один допускается новым предикатом. Сортировка весов в дереве BFS вводит порядок вершин дерева BFS в каждом слое — множестве вершин, находящихся на одинаковом расстоянии от корня дерева BFS.

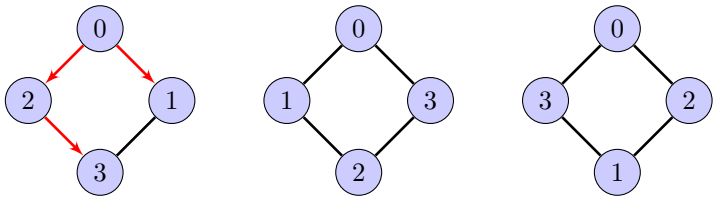

Рисунок 7 – Все графы, изоморфные C_4. Только левый граф занумерован в порядке BFS, но он имеет веса, которые не отсортированы по возрастанию.

Для кодирования нового ограничения в модель CSP необходимо добавить новые переменные, которые отвечают за веса поддеревьев каждой вершины: $\forall v\colon w_v$. Следующие ограничения определяют значения этих переменных (12) и накладывают ограничение $\varphi_{\text{BFS}}^{\deg+w}$ (13):

$$\forall u\colon w_u = 1 + \sum_{v=u+1}^{|V|-1} w_v \cdot [p_v = u], \tag{12}$$

$$\forall u\colon p_u = p_{u+1} \Rightarrow w_u \geqslant w_{u+1}. \tag{13}$$

Выражение $[a]$ равно единице, если выражение a истинно, и нулю в противном случае (в языке BEE для этого есть специальная команда `bool2int`).

2.5. Ограничение структуры графа

Все рассмотренные ранее методы использовали информацию только о дереве BFS (кроме ограничения стартовой вершины). Из-за этого ни один метод не может разделить набор графов с одинаковым деревом BFS. Пример такого набора графов изображен на рисунке 8. На нем изображены графы, каждый из которых допускается предикатом $\varphi_{\text{BFS}}^{\deg+w}$. Они все изоморфны, но информации о дереве BFS недостаточно, чтобы различить их. Для того, чтобы построить нарушение симметрии, которое запрещает больше изоморфных графов, необходимо рассматривать дополнительную информацию, помимо дерева BFS. Одним из предлагаемых подходов является учет степени вершины.

Определение 28. Предикат $\varphi_{\text{BFS}}^{\deg+w/d}(G) = 1$ тогда и только тогда, когда $\varphi_{\text{BFS}}^{\deg+w}(G) = 1$ и, дополнительно, если для пары детей вершины веса

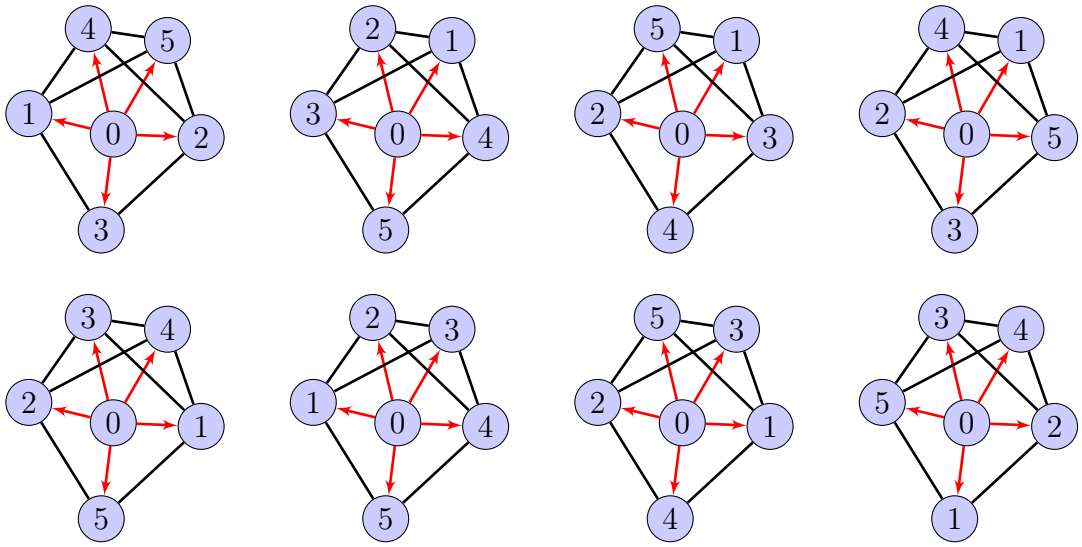

Рисунок 8 – Восемь изоморфных графов с совпадающими деревьями BFS. Каждый граф допускается предикатом $\varphi_{\text{BFS}}^{\deg+w}$.

поддеревьев равны, то они упорядочены по степени:

$$\forall\, v \in V(G) \colon \forall\, u < t \colon\ (p_u = v \wedge p_t = v \wedge w(u) = w(t)) \Rightarrow \deg u \geqslant \deg t \quad (14)$$

(дети отсортированы лексикографически по паре $(w(u), \deg u)$ вместо веса).

Теорема 4. Предикат $\varphi_{\text{BFS}}^{\deg+w/d}$ является корректным предикатом нарушения симметрии для множества связных графов.

Доказательство. Из доказательства теоремы 3 следует, что если поменять две вершины в одном слое местами, то вес левой не уменьшится, а правой — не увеличится. Рассмотрим две соседние в слое вершины u и v такие, что $w(u) = w(v)$ и $\deg u < \deg v$. Если поменять их местами, то степени вершин поменяются местами, вес u не увеличится, вес v не уменьшится: $w(u') \leqslant w(u)$, $w(v') \geqslant w(v)$, $\deg u' > \deg v'$. Отсюда $w(v') \geqslant w(v) = w(u) \geqslant w(u')$, то есть упорядоченность весов для этой пары вершин не изменилась, а степени вершин теперь стоят в правильном порядке.

Пусть существует пара вершин в одном слое, таких, что их веса одинаковы, а степени упорядочены по возрастанию. Тогда существует и пара таких соседних в слое вершин (иначе, по транзитивности и из-за упорядоченности вершин по весам, не существовало бы никакой такой пары вершин). Поменяем эти вершины местами, порядок весов не из-

менится, а степени вершин встанут в правильном порядке. Последовательно применяя такую операцию в итоге получим граф, который удовлетворяет $\varphi_{\text{BFS}}^{\deg+w/d}$ (процесс всегда закончится, потому что каждая операция строго увеличивает количество инверсий в последовательности степеней вершин в каждом слое, количество инверсий в слое не может превышать $a(a-1)/2$, где a — количество вершин в слое). □

Можно заметить, что конкретный порядок степеней (возрастающий или убывающий) не важен, оба варианта дают корректное нарушение симметрии.

Для кодирования этого ограничения в модель CSP новые переменные не понадобятся, так как степени и веса поддеревьев были введены для предыдущих предикатов. К ограничениям в модель CSP добавится ограничение (14).

2.6. Обобщение на несвязные графы

Предложенные методы могли применяться только к задачам поиска графов, решения которых являются связными графами. Однако есть много задач, в которых решением может быть несвязный граф. Например, задача о поиске реберной двухцветной раскраски может иметь в качестве ответа несвязный граф, потому что наличие ребра означает один цвет, а отсутствие — другой. Чтобы перебрать все возможные раскраски, необходимо перебрать все графы, в том числе несвязные. В данном разделе предлагается обобщение предложенных методов нарушения симметрии для связных графов, которое работает для нарушения симметрии в задаче поиска произвольных графов.

Основная идея, на которой основаны все предложенные нарушения симметрии, заключается в нумерации графа в порядке обхода в ширину. Несмотря на то, что несвязный граф нельзя полностью обойти обходом в ширину, можно предложить другое решение. Добавим к графу фиктивную вершину с номером -1. Для каждой компоненты связности произвольно выберем одну вершину и соединим ее ребром с фиктивной вершиной. Полученный граф будет связным, а значит в нем можно построить обход в ширину, проходящий по всем вершинам. На основе такой конструкции предлагается следующий предикат:

Определение 29. Предикат $\psi_{\text{BFS}}(G) = 1$ тогда и только тогда, когда в граф G можно добавить новую вершину v_{-1} и провести из нее по одному ребру в каждую компоненту связности таким образом, чтобы исходный граф был занумерован в порядке обхода BFS со стартовой вершиной v_{-1}.

Пример графа, удовлетворяющего ψ_{BFS}, а также соответствующего ему обхода BFS на дополненном графе изображен на рисунке 9

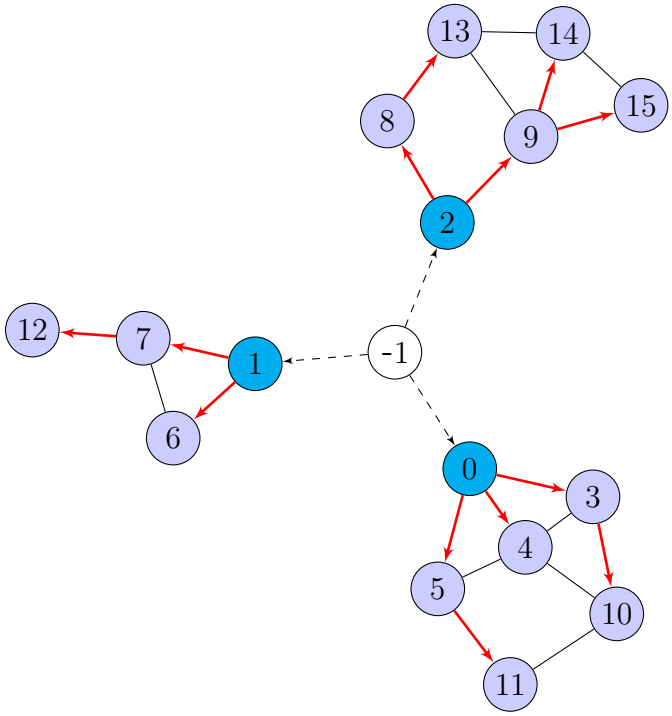

Рисунок 9 – Пример графа, допускаемого ψ_{BFS}. Пунктирными линиями изображены фиктивные ребра, красными стрелками — ребра деревьев BFS.

Теорема 5. Предикат ψ_{BFS} является корректным предикатом нарушения симметрии для произвольных графов.

Доказательство. Рассмотрим граф G. Добавим к нему вершину v_{-1} и проведем по одному ребру из нее в каждую компоненту связности. Полученный граф G' является связным, поэтому, по теореме 1 найдется изоморфный ему граф $\tilde{G}'$, который занумерован в порядке обхода в ширину (с началом в вершине v_{-1}, так как стартовая вершина может быть выбрана произвольно). Граф $\tilde{G}'$ изоморфен G', а вершина v_{-1} в обоих графах имеет одинаковый номер -1. Поэтому граф $\tilde{G}$, полученный из $\tilde{G}'$ удалением вершины v_{-1} будет изоморфен графу G, а также будет выполнено $\psi_{\text{BFS}}(\tilde{G}) = 1$ по построению. Следовательно, для любого гра-

фа найдется изоморфный ему граф, который удовлетворяет предикату ψ_{BFS}. □

Таким образом предикат ψ_{BFS} можно применять в задачах поиска произвольных графов. Однако, для прямого кодирования предложенного предиката в CSP пришлось бы ввести фиктивную вершину и дополнительные переменные матрице смежности, отвечающие за ее ребра. Гораздо удобнее использовать альтернативную модель, которая задает тот же самый предикат ψ_{BFS}. Заведем набор переменных p_i аналогично модели для предиката φ_{BFS}, $p_u = v$ означает, что вершина v является родителем вершины u в дереве BFS. Для вершин, которые являются корнями деревьев BFS в каждой компоненте связности положим $p_v = v$. При фиксировании родителя таким способом сохраняется свойство упорядоченности родителей (7), потому что для каждой компоненты связности номер корня — минимальный среди номеров вершин компоненты связности, поэтому он имеет такой же порядок относительно остальных вершин, как и номер настоящего родителя -1. Также определение $p_v = v$ для корней деревьев BFS позволяет сохранить ограничение (6) с единственным изменением — смена строгого знака неравенства на нестрогий:

$$\forall v : p_v \leqslant v. \tag{15}$$

Само определение родителей вершины (5) не изменится. Заметим, что если есть ребро в вершину v из вершины $u < v$, то будет выполнено $p_v = u$, как и для предиката φ_{BFS}. В случае, если не существует такого ребра, ни одно из значений $[0, v-1]$ не может быть равно p_v, поэтому p_v принимает единственное оставшееся значение $p_v = v$ (так как наложено ограничение $p_v \leqslant v$). Так происходит благодаря тому, что p_v определена как переменная целого типа в модели CSP, поэтому на множество значений $p_v = u$ будет наложено ограничение, которое фиксирует, что у переменной может быть ровно одно значение.

Аналогично нумерации в порядке обхода в ширину, ограничение на веса поддеревьев переносится на случай несвязных графов. В дополнение к ограничению ψ_{BFS} можно потребовать выполнения условия отсортированности весов для каждой компоненты связности. Более того, рассмотрение весов поддеревьев позволит частично устранить новый

вид симметрии, который возникает в случае несвязного графа. Помимо произвольной стартовой вершины (которых в случае несвязного графа стало несколько и произвольной является набор стартовых вершин) и произвольного порядка ребер для каждой вершины, для несвязного графа возникает еще один вид симметрии — порядок компонент связности может быть произвольным. То есть, если вершины $v \in [0, m-1]$ являются корнями деревьев BFS для каждой из m компонент связности графа, то существует $m!$ способов выбрать порядок этих компонент. Но рассмотрение весов поддеревьев может улучшить ситуацию. Рассмотрим вес поддерева с корнем в вершине v, где v — корень дерева BFS для целой компоненты связности. Тогда w_v равняется количеству вершин в данной компоненте связности. На основе этой информации предлагается дополнительно ограничить компоненты связности таким образом, чтобы они были упорядочены по размеру. На основе всего вышесказанного предлагается следующий предикат:

Определение 30. Предикат $\psi^w_{\text{BFS}}(G) = 1$ тогда и только тогда, когда $\psi_{\text{BFS}}(G) = 1$, для каждой вершины v веса поддеревьев ее детей упорядочены в порядке невозрастания, а веса корней деревьев BFS для каждой компоненты связности также упорядочены в порядке невозрастания.

Теорема 6. Предикат ψ^w_{BFS} является корректным предикатом нарушения симметрии для произвольных графов.

Доказательство. Так как граф G удовлетворяет предикату ψ_{BFS}, то можно добавить вершину и ребра в некоторый изоморфный граф $\tilde{G}$ так, чтобы полученный граф был занумерован в порядке обхода BFS. По теореме 3 для такого расширенного графа найдется изоморфный граф $\tilde{G}'$, для которого будет выполнено условие отсортированности весов. Заметим, что это означает, что для каждой вершины исходного графа выполняется условие отсортированности весов. А условие отсортированности весов для новой вершины -1 эквивалентно упорядоченности весов исходных компонент связности. □

На рисунке 10 изображен пример графа, изоморфного графу с рисунка 9 и удовлетворяющий предикату ψ^w_{BFS}. Компоненты связности со стартовыми вершинами $0, 1$ и 2 имеют количество вершин $6, 6$ и 4 соот-

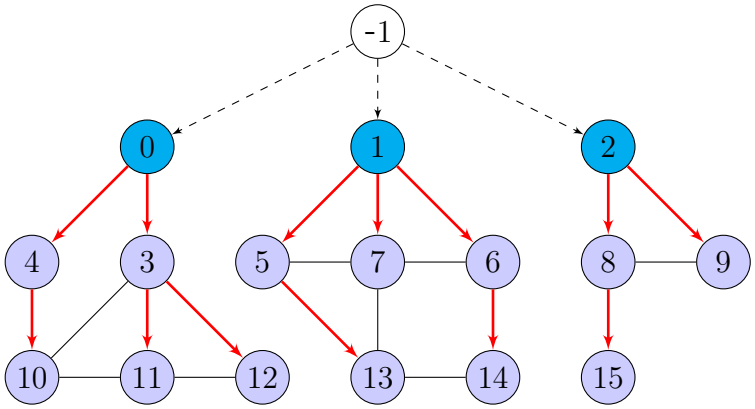

Рисунок 10 – Пример графа, допускаемого ψ^{w}_{BFS}, изоморфный графу на рисунке 9. Пунктирными линиями изображены фиктивные ребра, красными стрелками — ребра деревьев BFS.

ветственно. Дерево BFS в каждой отдельной компоненте связности тоже имеет упорядоченные веса поддеревьев.

Для кодирования предиката ψ^{w}_{BFS} в модель CSP используются уже описанные ранее для предиката $\varphi^{\deg+w}_{\text{BFS}}$ определение весов (12) и условие упорядоченности (13). К этому добавляется новое ограничение, задающее порядок следования компонент связности:

$$\forall i, j; i < j : (p_i = i \wedge p_j = j) \Rightarrow w_i \geqslant w_j \tag{16}$$

Выводы по главе 2

В отличие от конечных автоматов, нарушение симметрии на основе обхода BFS для графов устраняет гораздо меньше симметрий. Это происходит из-за того, что в автомате существует выделенное стартовое состояние, а также введен порядок на ребрах (символы перехода). Несмотря на это, был предложен набор дополнительных ограничений, который уменьшает разницу в эффективности этих нарушений симметрии между графами и конечными автоматами. Для небольших графов такие методы устраняют все симметрии для большого числа классов изоморфизма. Также предложенные методы вводят степени вершин, что может быть полезно для графовых задач, в которых накладываются ограничения на степень.

ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ

В данной главе описываются проведенные эксперименты, которые сравнивают эффективность применения методов нарушения симметрий, предложенных в данной работе, с методами, предложенными в работе [2] (sb_ℓ и sb^*_ℓ) и в работе [5] (USG-SBP).

3.1. Нахождение $ex(n; C_3, C_4)$

Задача о нахождении максимального количества ребер в графе из n вершин с обхватом не меньше пяти является классической задачей комбинаторной оптимизации. Одним из методов решения таких задач является перебор ответа и проверка, существует ли хотя бы один объект с заданными свойствами. На основе этого подхода формулируется следующая задача поиска графов: необходимо найти граф, имеющий n вершин, m ребер и не содержащий циклов длины 3 и 4.

3.1.1. Базовая модель

Граф задается матрицей смежности ограничениями (3) и (4). Для того, чтобы построить систему из $O(n^3)$ ограничений, которые запрещают циклы длины 3 и 4 вводятся дополнительные переменные, как в работе [2]. Переменные $x_{ijk}, i < k$ соответствуют существованию пути $i - j - k$:

$$\forall\, i, j, k; i < k \colon x_{ijk} \Leftrightarrow A[i, j] \wedge A[j, k]. \tag{17}$$

Переменные $x_{ik}, i < k$ соответствуют существованию произвольного пути длины 2 между вершинами i и k:

$$\forall\, i, k; i < k \colon x_{ik} \Leftrightarrow \bigvee_{\substack{j=0 \\ j \neq i, j \neq k}}^{n-1} x_{ijk}. \tag{18}$$

С использованием переменных $\{x_{ik}\}$ и $\{x_{ijk}\}$ можно сформулировать следующие ограничения, запрещающие циклы длины 3 и 4 (ограничения (19) и (20)):

$$\forall\, i, k; i < k \colon \neg(A[i, k] \wedge x_{ik}), \tag{19}$$

$$\forall\, i, k; i < k \colon \sum_{j=i+1}^{n-1} x_{ijk} < 2. \tag{20}$$

Полученная модель полностью описывает поставленную задачу, однако, добавление дополнительных ограничений даст больше информации, с помощью которой возможно уменьшение пространства поиска.

В работе [15] доказано, что следующие свойства выполнены для экстремальных графов (графов из $EX(n; C_3, C_4)$):

$$n \geqslant 1 + \Delta\delta, \tag{21}$$

$$\delta \geqslant m - ex(n-1; C_3, C_4), \tag{22}$$

$$\Delta \geqslant \lceil 2m/n \rceil, \tag{23}$$

где δ — минимальная степень вершины, а Δ — максимальная.

Соотношения (21) и (22) специфичны для поставленной задачи, а свойство (23) выполнено для любого графа, так как максимальная степень вершины не меньше, чем средняя степень среди всех вершин в графе. Дополнительно в качестве ограничений накладываются следующие свойства, верные для любого графа:

$$\delta \leqslant \Delta, \tag{24}$$

$$\forall v \colon \delta \leqslant \deg v \leqslant \Delta, \tag{25}$$

$$\sum_{v \in V} \deg v = 2m. \tag{26}$$

На этом описание базовой модели закончено, все переменные этой модели представлены в таблице 1, а ограничения — в таблице 2.

Таблица 1 – Переменные в модели для нахождения $ex(n; C_3, C_4)$

Имя	Множество значений	Индексы
A_{ij}	$[0, 1]$	$i, j \in [0, n-1]$
x_{ijk}	$[0, 1]$	$i, j \in [0, n-1], k \in [i+1, n-1]$
x_{ik}	$[0, 1]$	$i \in [0, n-1], k \in [i+1, n-1]$
$\deg_i$	$[1, n-1]$	$i \in [0, n-1]$
max_deg	$[1, n-1]$	–
min_deg	$[1, n-1]$	–

Таблица 2 – Ограничения в модели для нахождения $ex(n; C_3, C_4)$

Ограничение	Индексы
$A_{ij} = A_{ji}$	$i, j \in [0, n-1]$
$A_{ii} = \text{false}$	$i \in [0, n-1]$
$x_{ijk} = A_{ij} \wedge A_{jk}$	$i, j, k \in [0, n-1], i < k$
$x_{ik} = \bigvee_{j=0}^{n-1} x_{ijk}$	$i, k \in [0, n-1], i < k$
$\neg A_{ik} \vee \neg x_{ik}$	$i, k \in [0, n-1], i < k$
$\sum_{j=0}^{n-1} x_{ijk} < 2$	$i, k \in [0, n-1], i < k$
$\deg_i = \sum_{j=0}^{n-1} A_{ij}$	$i \in [0, n-1]$
$\min\_\deg = \min_{j \in [0,n-1]} \deg_j$	–
$\max\_\deg = \max_{j \in [0,n-1]} \deg_j$	–
$\min\_\deg \leqslant \max\_\deg$	–
$\max\_\deg \geqslant \lceil 2m/n \rceil$	–
$\deg_i \leqslant \max\_\deg$	$i \in [0, n-1]$
$\deg_i \geqslant \min\_\deg$	$i \in [0, n-1]$
$\min\_\deg \cdot \max\_\deg \leqslant n-1$	–
$\min\_\deg \geqslant m - ex(n-1; C_3, C_4)$	–
$\sum_{i=0}^{n-1} \deg_i = 2m$	–

3.1.2. Нарушения симметрии

На основе описанной базовой модели строятся новые путем добавления описанных в этой работе предикатов нарушения симметрии. Заметим, что экстремальные решения поставленной задачи всегда являются связными графами.

Утверждение 2. $G \in EX(n; C_3, C_4) \Rightarrow G$ — связный.

Доказательство. Пусть не так: существует граф $G \in EX(n; C_3, C_4)$, которые имеет m ребер и несвязный. Возьмем две вершины $u, v \in V(G)$ такие, что не существует пути в графе между ними (вершины из разных компонент связности). Добавим ребро $\langle u, v \rangle$ в граф G и получим новый граф G'. В графе G' ребер на одно больше, чем $ex(n; C_3, C_4)$, значит в нем есть цикл длины 3 или 4, так как $ex(n; C_3, C_4)$ — максимальное число ребер в графе из n вершин без циклов длины 3 и 4. Этот цикл обязательно содержит ребро $\langle u, v \rangle$, так как в исходном графе G не было таких циклов. Значит, вершины u и v лежат на цикле, а значит существует хотя бы два пути, соединяющие эти вершины, и хотя бы один из таких путей не содержит ребра $\langle u, v \rangle$. Значит, убрав ребро из графа G', в новом графе

(который совпадает с G) вершины u и v будут связаны каким-то путем, что противоречит условию их выбора. Следовательно, такого графа G не существует. □

Благодаря этому свойству экстремальных графов к данной задаче применимы описанные в работе методы нарушения симметрии для связных графов.

Для проведения экспериментов строятся 7 различных моделей: базовая ($\varnothing$), с BFS-нумерацией (φ_{BFS}), с дополнительным ограничением стартовой вершины ($\varphi_{\text{BFS}}^{\deg}$), с ограничением структуры BFS-дерева ($\varphi_{\text{BFS}}^{\deg+w}$), с дополнительным ограничением структуры графа ($\varphi_{\text{BFS}}^{\deg+w/d}$), а также с использованием предикатов нарушения симметрии из работ [2] (sb_{ℓ}^{*}) и [5] (USG-SBP).

3.1.3. Описание экспериментов

Каждая модель, полученная в предыдущих разделах, с помощью BEE [12] транслируется в SAT. Полученная задача решается с помощью программного средства treengeling [7] с ограничением в 4 потока и 16 Гб оперативной памяти на процессоре AMD Opteron 6378 @ 2.4 GHz. Ограничение по времени решения составляет один час (3600 секунд). Все эксперименты были повторены с использованием программного средства glucose-syrup в аналогичных условиях. Все полученные значения отличались не более, чем на 15% и сохранялся относительный порядок измерений для разных моделей.

Случай UNSAT (отсутствие решения) является более стабильным по времени, потому при решении в любом случае необходимо обойти все пространство поиска целиком. Поэтому при измерении этого случая проводилось только 5 измерений времени для каждой модели и вычислялось среднее значение (так как отклонение от среднего составляло менее 1%). В случае SAT (существования решения), из-за наличия нескольких решений задачи и того, что современные программные средства для решения задачи SAT используют случайность в процессе поиска, время решения задачи может существенно отличаться (от десятков секунд до превышения ограничения по времени). Из-за этого, для измерения более достоверных значений, время решения каждой модели измерялось 50 раз.

Для полученной последовательности из 50 измерений вычислялись статистики: три квартиля ($0{,}25$-квантиль, медиана и $0{,}75$-квантиль). Эти значения имеют большее практическое значение, чем среднее значение. В случае, когда время решения модели может существенно различаться, используется метод портфолио: запускается несколько программ решения параллельно и ожидается, пока любая из них выдаст ответ. В таком случае время решения модели составляет минимум нескольких независимых измерений. Так как медиана — это такая статистика, что случайная величина превышает ее с вероятностью 50%, то минимум из k измерений будет превышать медиану с вероятностью $(100/2^k)\,\%$. Благодаря этому свойству медиану можно использовать, чтобы вычислить, какое число программ необходимо запустить параллельно, чтобы найти решение быстрее, чем за медиану, с заданной вероятностью.

На рисунках 11 и 12 изображены времена работы для случаев $m = ex(n; C_3, C_4)$ и $m = ex(n; C_3, C_4) + 1$ соответственно. Для случая отсутствия решения модель без нарушения симметрии и с предикатом USG-SBP не решала ни одну задачу при $n > 12$ за отведенное время, поэтому они не изображены на графике. Для случая существования решения метод USG-SBP не давал существенного выигрыша по сравнению с отсутствием нарушений симметрии, поэтому его результат не представлен на графике.

Все результаты измерений приведены в таблицах 3 и 4. В таблице приведена медиана (в секундах) для каждой серии измерений. Символ «—» означает превышение ограничения времени, установленного в 3600 секунд (один час). Для случая удовлетворимости каждая серия экспериментов содержала 50 измерений, для случая отсутствия решения — пять. Для случая отсутствия решения treengeling запускался с дополнительной опцией «`--no-simp`», которая отключает статический оптимизатор входного файла. Для больших моделей это позволило сэкономить существенное количество времени.

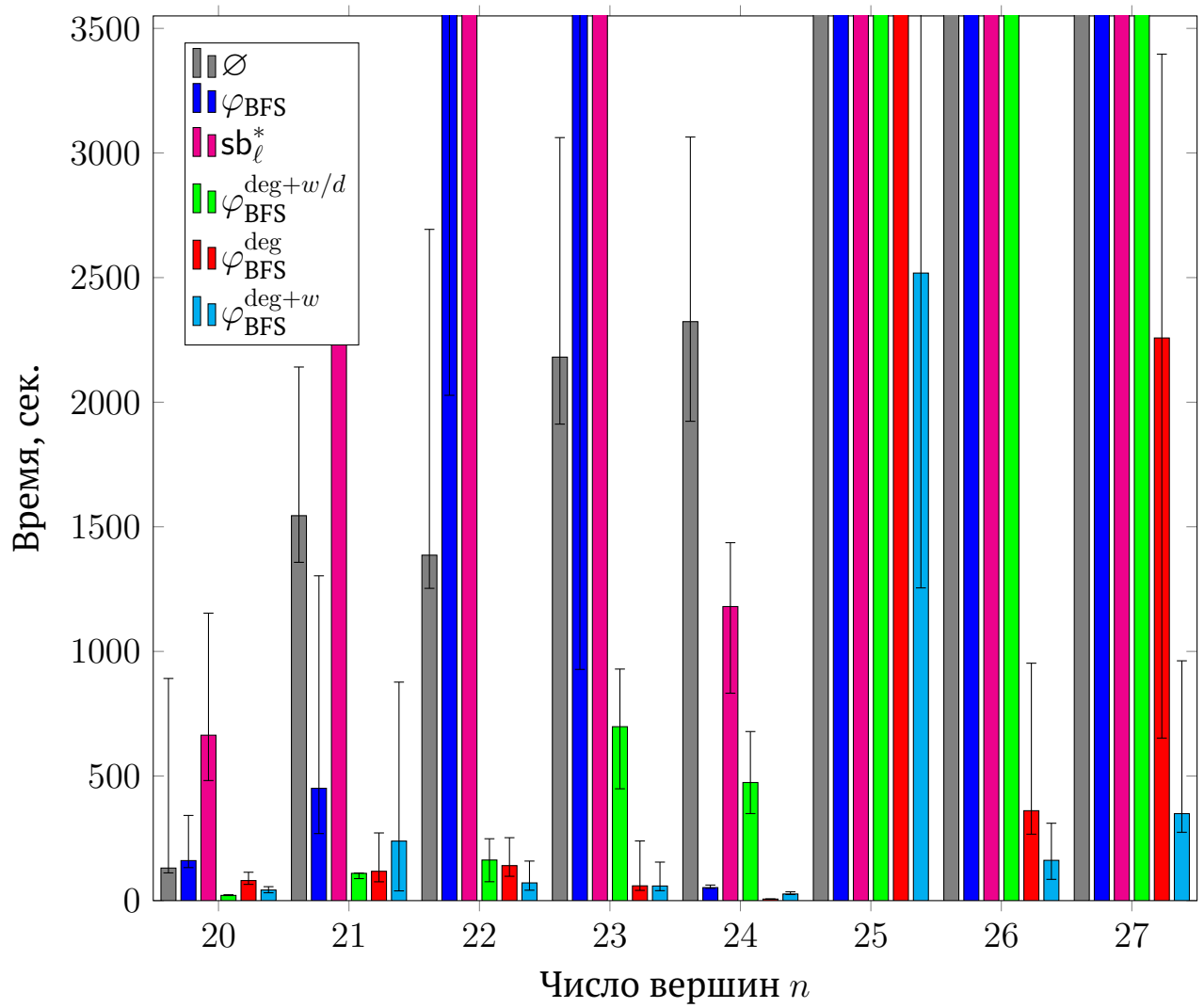

Рисунок 11 – Время решения моделей при $m = ex(n; C_3, C_4)$, статистика по 50 замерам

Таблица 3 – Время нахождения решения для моделей в случае существования решения $(m = ex(n; C_3, C_4))$.

n	$\varnothing$	φ_{BFS}	$\varphi^{\deg}_{\text{BFS}}$	$\varphi^{\deg+w}_{\text{BFS}}$	$\varphi^{\deg+w/d}_{\text{BFS}}$	sb^*_ℓ	USG_SBP
18	5,68	**1,04**	1,08	4,12	3,00	10,38	14,37
19	0,98	**0,92**	1,08	1,35	1,34	1,37	1,01
20	126,10	344,95	77,81	42,51	**21,17**	1234,16	2517,60
21	985,73	617,24	115,89	239,00	**109,36**	2974,09	3528,72
22	2139,65	455,36	141,15	**71,21**	75,43	—	—
23	—	—	63,54	**58,71**	760,84	—	—
24	1765,81	52,30	**6,22**	26,64	492,64	1237,63	—
25	—	—	—	**2518,18**	—	—	—
26	—	—	433,77	**161,68**	—	—	—
27	—	—	1982,06	**349,10**	—	—	—

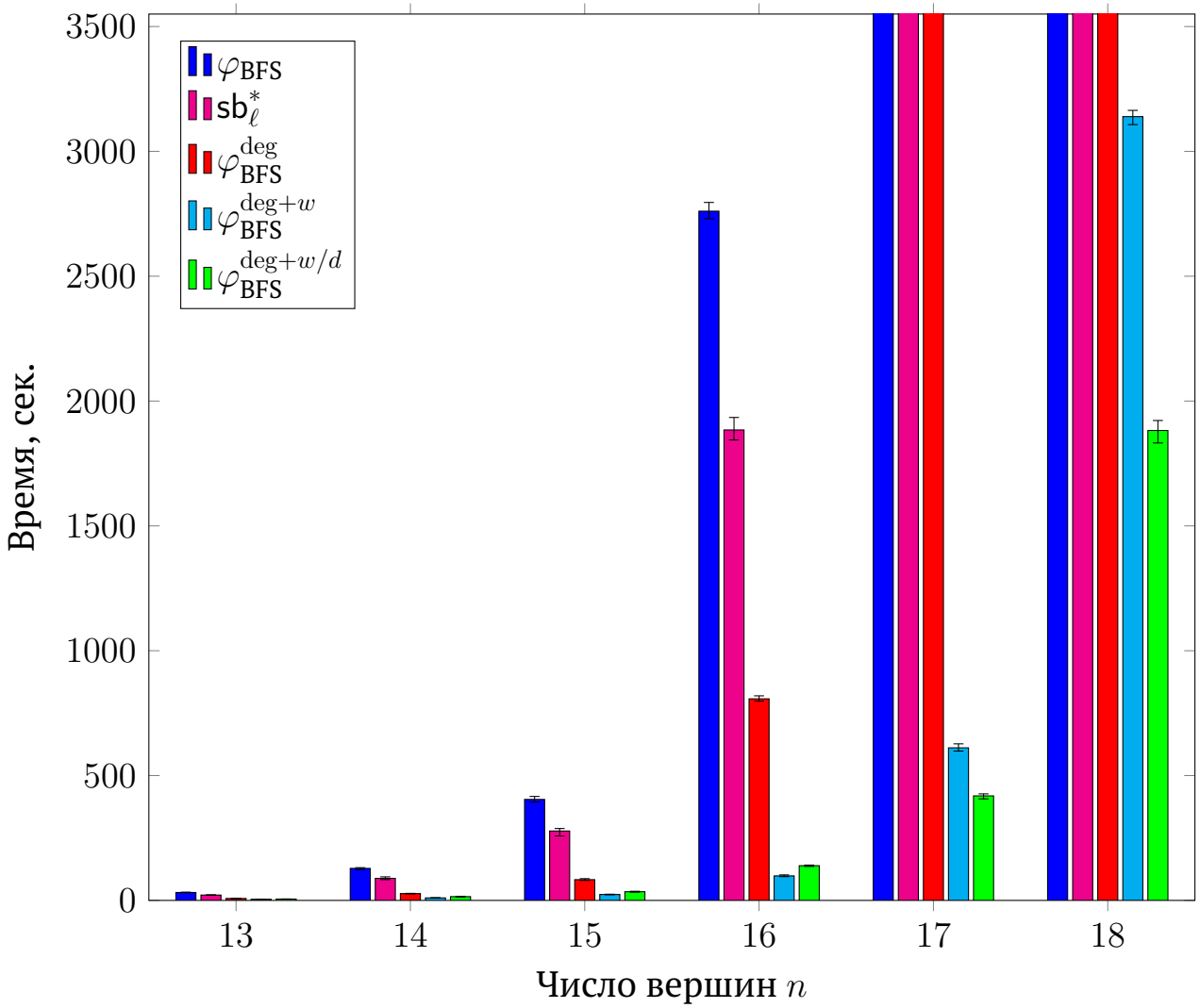

Рисунок 12 – Время решения моделей при $m = ex(n; C_3, C_4) + 1$, статистика по 5 замерам

Таблица 4 – Время нахождения решения для моделей в случае отсутствия решения $(m = 1 + ex(n; C_3, C_4))$.

n	∅	φ_{BFS}	$\varphi^{\deg}_{\text{BFS}}$	$\varphi^{\deg+w}_{\text{BFS}}$	$\varphi^{\deg+w/d}_{\text{BFS}}$	sb^*_ℓ	USG_SBP
10	66,98	0,59	**0,26**	0,43	0,30	0,35	n/a[1]
11	2572,06	3,03	0,79	**0,40**	**0,40**	1,28	n/a[1]
12	—	5,53	2,41	**1,05**	1,39	5,01	2368,04
13	—	17,79	5,76	**4,23**	4,32	14,80	—
14	—	58,63	19,36	8,82	**8,30**	65,97	—
15	—	279,17	76,81	21,82	**20,05**	254,83	—
16	—	2408,45	597,59	95,28	**65,43**	1455,55	—
17	—	—	—	514,81	**304,23**	—	—
18	—	—	—	2773,63	**1694,09**	—	—
19	—	—	—	—	**3576,24**	—	—

3.2. Нахождение $ex(n; C_4)$

Еще одной задачей из класса задач Турана является задача о нахождении максимального числа ребер в графе из n вершин, который не содержит циклов длины 4 (граф, свободный от четырехугольников, quadrilateral-free graph). Она рассматривалась в ряде работ, например в [13] и [14]. Также такие графы исследовались с достаточно неожиданной стороны в работе [16]. Точное решение этой задачи известно только для чисел специального вида: если $n = q^2 + q + 1$, где $q = 2^k$ или $q = p^k$, p — простое, $p > 13$, тогда $ex(n; C_4) = \frac{1}{2}q(q+1)^2$.

Для данной задачи известно меньше ограничений, которые помогли бы во время поиска таких графов. Модель для новой задачи строится на основе модели для нахождения $ex(n; C_3, C_4)$, из которой убираются ограничения 19 (отсутствие циклов длины 3), 21 и 22 (специфичные свойства графов из $EX(n; C_3, C_4)$). Все остальные ограничения остаются неизменными.

Эксперименты были полностью аналогичны экспериментам для задачи нахождения $ex(n; C_3, C_4)$, включая ограничение по времени, средство решения задачи SAT и количество экспериментов в серии (50 измерений для случая существования решения, 5 измерений в случае неудовлетворимости модели). Для сравнения были построены модели, аналогичные описанным ранее — базовая модель и шесть моделей с добавлением предикатов нарушения симметрии: φ_{BFS}, $\varphi^{\deg}_{\text{BFS}}$, $\varphi^{\deg+w}_{\text{BFS}}$, $\varphi^{\deg+w/d}_{\text{BFS}}$, sb^*_ℓ и USG-SBP. Все результаты измерений приведены в таблицах 5 и 6. В таблице приведена медиана (в секундах) для каждой серии измерений. Символ «—» означает превышение ограничения времени, установленного в 3600 секунд (один час).

3.3. Нахождение $R(l, k)$

Среди задач, связанных с раскраской графа, одной из самых известных является задача о поиске чисел Рамсея. Ее двухцветная формулировка основывается на частном случае теоремы Рамсея:

Утверждение 3 (Двухцветный вариант теоремы Рамсея). Для любых чисел l, k найдется такое число $R(l, k)$, что для любой раскраски ребер

[1]Не применимо, так как использовался неизбежный подграф для K_{12}, поэтому не является предикатом нарушения симметрии для графовых задач с количеством вершин меньше 12.

Таблица 5 – Время нахождения решения для моделей в случае существования решения ($m = ex(n; C_4)$).

n	$\varnothing$	φ_{BFS}	$\varphi^{\deg}_{\text{BFS}}$	$\varphi^{\deg+w}_{\text{BFS}}$	$\varphi^{\deg+w/d}_{\text{BFS}}$	sb^*_ℓ	USG_SBP
17	**42,50**	297,69	280,39	57,72	98,41	205,02	55,24
18	1277,22	1143,39	1279,65	**258,81**	1808,90	1500,15	3425,28
19	**111,94**	742,81	1445,58	292,67	383,75	3422,95	546,35
20	—	—	—	**1489,93**	—	—	—
21	—	—	—	**2548,04**	—	—	—
22	—	—	—	**2796,53**	—	—	—

Таблица 6 – Время нахождения решения для моделей в случае отсутствия решения ($m = 1 + ex(n; C_4)$).

n	$\varnothing$	φ_{BFS}	$\varphi^{\deg}_{\text{BFS}}$	$\varphi^{\deg+w}_{\text{BFS}}$	$\varphi^{\deg+w/d}_{\text{BFS}}$	sb^*_ℓ	USG_SBP
8	4,50	0,34	0,54	0,22	0,23	**0,21**	n/a
9	49,93	0,95	0,53	0,30	**0,29**	0,43	n/a
10	581,05	2,20	1,11	**0,51**	0,73	0,92	n/a
11	—	7,56	2,84	2,26	**1,59**	3,68	n/a
12	—	23,21	12,48	6,29	**5,72**	12,73	—
13	—	94,32	33,80	12,95	**11,82**	43,07	—
14	—	620,49	172,32	30,45	**27,42**	233,13	—
15	—	3285,48	768,11	61,46	**56,51**	2008,36	—
16	—	—	—	311,48	**200,41**	—	—
17	—	—	—	2181,37	**1800,79**	—	—

графа $K_{R(l,k)}$ в красный и синий цвета найдется либо полный красный подграф из l вершин, либо полный синий подграф из k вершин.

Для нахождения чисел $R(l, k)$ удобно пользоваться понятием (n, l, k)-раскраски:

Определение 31. (n, l, k)-раскраской называется такая раскраска графа K_n, которая не содержит полного красного подграфа из l вершин и не содержит полного синего подграфа из k вершин.

С использованием этого определения можно сформулировать задачу поиска чисел Рамсея как задачу поиска графа: $R(l, k) = r$ тогда и только тогда, когда не существует (r, l, k)-раскраски. Заметим, что двуцветная раскраска полного графа эквивалентна обычному графу, где ребро в графе обозначает красное ребро в раскраске, а отсутствие ребра — синее.

3.3.1. Модель CSP

Граф задается матрицей смежности ограничениями (3) и (4). Для задания раскраски используется ограничение из работы [22]:

$$\bigwedge_{c\in[0,1]}\bigwedge_{I\in\wp_r(V(K_r))}\bigvee\{A_{ij}\neq c\mid i,j\in I, i<j\}, \tag{27}$$

где $\wp_r(V(K_r))$ — множество всех подмножеств размера r вершин графа, а c пробегает возможные цвета (0 — красный, 1 — синий).

К данной модели применяются предложенные предикаты нарушения симметрии ψ_{BFS} и ψ^w_{BFS}, а также предикат sb^*_ℓ из работы [2].

3.3.2. Результаты экспериментов

Эксперименты проводились на том же оборудовании, как и для предыдущих задач. Ограничение по времени составляло 3600 секунд (один час). Проводились измерения для $r < R(l,k)$ (случай существования искомой раскраски) и для $r \geqslant R(l,k)$ (случай отсутствия искомой раскраски). Измерения показали, что время решения модели отличалось незначительно от запуска к запуску, поэтому для каждой модели была проведена серия из 10 измерений, по результату которой была вычислена медиана. Результаты измерений приведены в таблицах 7 и 8. В таблицах приведено медианное время решения моделей в секундах. Символ «—» означает превышение ограничения по времени. Пример найденного решения приведен на рисунке 13.

Выводы по главе 3

Результаты экспериментов показывают, что использование методов, основанных на обходе в ширину, во многих ситуациях эффективнее существовавших ранее методов. Метод $\varphi^{\deg+w}_{\text{BFS}}$ показал существенный прирост к скорости поиска решения, особенно для случая существования решения. Метод, основанный на предикате $\varphi^{\deg+w/d}_{\text{BFS}}$ дает преимущество только для случая отсутствия решения, существенно проигрывая для случая удовлетворимости модели. Одним из объяснений этого может быть то, что стратегия обхода пространства поиска, которую использовал treengeling, конфликтует с методом, так, что удовлетворяющие решения располагаются ближе к концу обхода пространства поиска.

Таблица 7 – Время нахождения решения для моделей в случае существования решения ($r < R(l,k)$).

(r,l,k)	$\varnothing$	ψ_{BFS}	ψ^{w}_{BFS}	sb^{*}_{ℓ}	USG_SBP
$(16,4,4)$	**0,1**	1,0	1,0	1,8	**0,1**
$(17,4,4)$	**0,6**	3,6	2,8	5,3	0,8
$(16,3,6)$	1,6	2,1	2,1	1,9	**1,4**
$(17,3,6)$	**1,7**	2,4	2,4	2,2	**1,7**
$(23,4,5)$	11,1	9,2	**9,0**	20,5	11,0
$(24,4,5)$	289,0	91,0	**90,0**	146,0	453,0
$(32,5,5)$	12,5	33,9	148,0	150,0	**11,9**
$(33,5,5)$	213,0	158,0	158,0	729,0	**127,3**

Таблица 8 – Время нахождения решения для моделей в случае отсутствия решения ($r \geqslant R(l,k)$).

(r,l,k)	$\varnothing$	ψ_{BFS}	ψ^{w}_{BFS}	sb^{*}_{ℓ}	USG_SBP
$(9,3,4)$	0,8	**0,1**	**0,1**	**0,1**	n/a
$(10,3,4)$	0,7	**0,1**	**0,1**	**0,1**	n/a
$(14,3,5)$	92,0	**0,3**	**0,3**	0,5	3,9
$(15,3,5)$	66,7	**0,4**	**0,4**	**0,4**	**0,4**
$(18,3,6)$	—	17,6	17,5	**5,0**	—
$(19,3,6)$	—	16,3	16,3	**4,0**	—
$(18,4,4)$	—	**4,5**	4,8	5,6	—
$(19,4,4)$	—	6,6	7,7	**5,5**	—
$(20,4,4)$	—	4,1	**4,0**	6,3	—
$(21,4,4)$	—	5,8	**4,9**	7,2	—
$(22,4,4)$	—	7,5	**7,4**	10,4	—
$(23,4,4)$	—	**3,4**	**3,4**	7,4	—

Из-за этого приходилось обойти почти все пространство поиска, чтобы добраться до решения. Это существенное отличие удовлетворимого случая задачи — в зависимости от расположения решения в пространстве поиска время поиска может отличаться на несколько порядков. В случае отсутствия решения, необходимо в любом случае обойти пространство поиска целиком, что делает такие ситуации стабильными по времени,

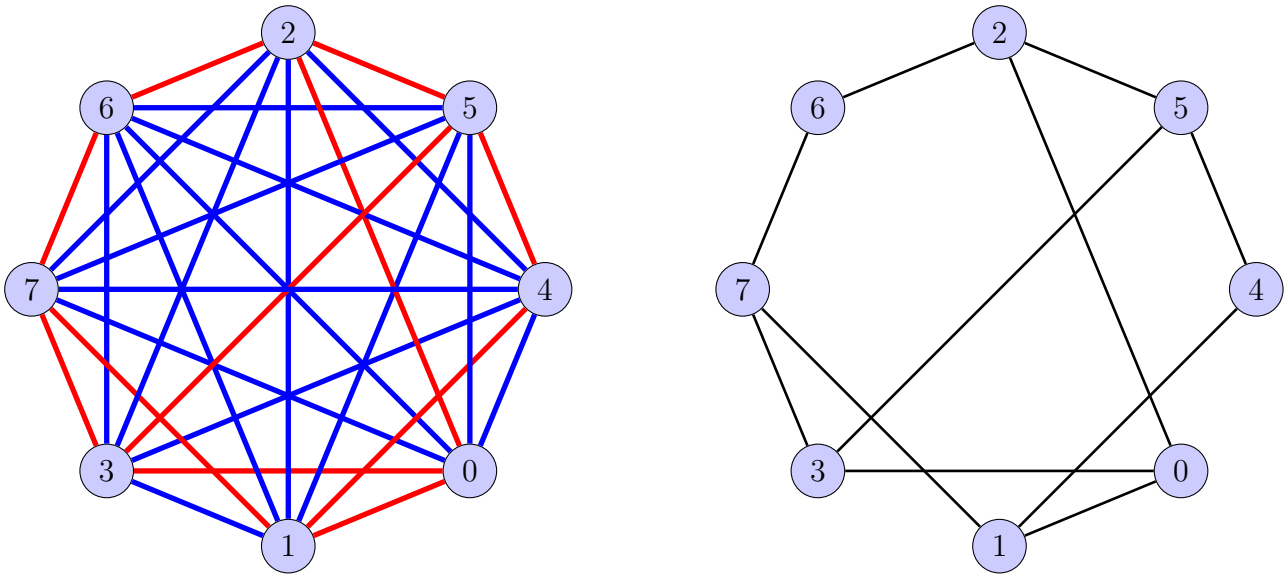

Рисунок 13 – Пример $(8, 3, 4)$-раскраски (слева) и соответствующего ей графа (справа).

применение методов нарушения симметрии обычно уменьшает время обхода пространства поиска в таком случае.

Предикаты для задачи поиска несвязных графов продемонстрировали эффективность, сравнимую с существующими методами, оказываясь лучше $\mathtt{sb}^*_\ell$ в большинстве ситуаций, однако в ряде случаев проигрывая USG-SBP. Последний продемонстрировал свое преимущество для малых графов, полностью выходя за рамки выделенного времени на больших размерах задачи. Таким образом, предложенные предикаты ψ_{BFS} и ψ^w_{BFS} показали себя эффективнее предложенных ранее методов, начиная с некоторого размера задачи.

Программу, реализующую все описанные эксперименты, можно найти по адресу `https://github.com/slavam2605/SymmetryBreakingMethods`.

ЗАКЛЮЧЕНИЕ

В данной работе были исследованы существующие методы нарушения симметрий в задачах поиска неориентированных графов. Были предложены четыре новых набора ограничений, нарушающих симметрии в задачах поиска связных графов. Было предложено два обобщения предложенных предикатов для задач поиска несвязных графов. Были предложены формальные доказательства корректности каждого из предложенных методов. Эти методы были применены к реальным задачам комбинаторной оптимизации и было проведено сравнение эффективности предложенных методов с существующими методами, один из которых на данный момент считается наиболее эффективным методом из всех известных на данный момент.

Было продемонстрировано, что для задач поиска связных графов два из предложенных методов ($\varphi_{\text{BFS}}^{\deg+w}$ и $\varphi_{\text{BFS}}^{\deg+w/d}$) обеспечивают заметно более высокую эффективность решения предложенных задач по сравнению с существующими методами. Метод, использующий предикат $\varphi_{\text{BFS}}^{\deg+w}$ показал высокую эффективность на обоих классах задач (удовлетворимый и неудовлетворимый случаи), а метод $\varphi_{\text{BFS}}^{\deg+w/d}$ был самым эффективным для ситуации отсутствия решения. Это делает эти два метода самыми эффективными методами нарушения симметрии для задач поиска связных графов среди предложенных на данный момент.

В случае с предикатами для задачи поиска несвязных графов, предложенные методы ψ_{BFS} и ψ_{BFS}^{w} продемонстрировали свое преимущество на задачах, начиная с некоторого размера. Таким образом предложенные методы работают эффективнее для больших задач поиска несвязных графов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Balyo T.*, *Heule M. J.*, *Järvisalo M.* SAT COMPETITION 2016. — 2016.

2 Breaking Symmetries in Graph Representation / M. Codish [et al.] // IJCAI. — 2013. — P. 3–9.

3 *Heule, Marijn JH* The Quest for Perfect and Compact Symmetry Breaking for Graph Problems // Theory and Applications of Satisfiability Testing – SAT 2016. — Springer, 2016. — P. 228–245.

4 *Ulyantsev V.*, *Zakirzyanov I.*, *Shalyto A.* BFS-based symmetry breaking predicates for DFA identification // International Conference on Language and Automata Theory and Applications. — Springer. 2015. — P. 611–622.

5 *Cuong C.*, *Heule M.* Computing Maximum Unavoidable Subgraphs Using SAT Solvers // International Conference on Theory and Applications of Satisfiability Testing. — Springer. 2016. — P. 196–211.

6 *Itzhakov A.*, *Codish M.* Breaking symmetries in graph search with canonizing sets // Constraints. — 2016. — Vol. 21, no. 3. — P. 357–374.

7 *Biere A.* Splatz, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2016 // SAT COMPETITION 2016. — P. 44.

8 *Audemard G.*, *Simon L.* Glucose in the SAT 2014 Competition // SAT COMPETITION 2014. — 2014. — P. 31.

9 *Soos M.* The CryptoMiniSat 5 set of solvers at SAT Competition 2016 // SAT COMPETITION 2016. — 2016. — P. 28.

10 *Jussien N.*, *Rochart G.*, *Lorca X.* Choco: an open source java constraint programming library // CPAIOR'08 Workshop on Open-Source Software for Integer and Contraint Programming (OSSICP'08). — 2008. — P. 1–10.

11 MiniZinc: Towards a standard CP modelling language / N. Nethercote [et al.] // International Conference on Principles and Practice of Constraint Programming. — Springer. 2007. — P. 529–543.

12 *Metodi A.*, *Codish M.* Compiling finite domain constraints to SAT with BEE // Theory and Practice of Logic Programming. — 2012. — Vol. 12, no. 4–5. — P. 465–483.

13 *Füredi Z.* Graphs without quadrilaterals // Journal of Combinatorial Theory, Series B. — 1983. — Vol. 34, no. 2. — P. 187–190.

14 *Füredi Z.* Quadrilateral-free graphs with maximum number of edges // preprint. — 1994.

15 *Garnick D. K.*, *Kwong Y.*, *Lazebnik F.* Extremal graphs without three-cycles or four-cycles // Journal of Graph Theory. — 1993. — Vol. 17, no. 5. — P. 633–645.

16 *De Freitas M. A. A.*, *Nikiforov V.*, *Patuzzi L.* Maxima of the Q-index: forbidden 4-cycle and 5-cycle // arXiv preprint arXiv:1308.1652. — 2013.

17 *Gandham S.*, *Dawande M.*, *Prakash R.* Link scheduling in wireless sensor networks: Distributed edge-coloring revisited // Journal of Parallel and Distributed Computing. — 2008. — Vol. 68, no. 8. — P. 1122–1134.

18 Register allocation via coloring / G. J. Chaitin [et al.] // Computer languages. — 1981. — Vol. 6, no. 1. — P. 47–57.

19 *Briggs P.*, *Cooper K. D.*, *Torczon L.* Improvements to graph coloring register allocation // ACM Transactions on Programming Languages and Systems (TOPLAS). — 1994. — Vol. 16, no. 3. — P. 428–455.

20 Breaking Symmetries in Graph Coloring Problems with Degree Matrices: the Ramsey Number R (4, 3, 3)= 30 / M. Codish [et al.] // arXiv preprint arXiv:1409.5189. — 2014.

21 Solving graph coloring problems with abstraction and symmetry / M. Codish [et al.] // arXiv preprint arXiv:1409.5189. — 2014.

22 Computing the ramsey number R(4, 3, 3) using abstraction and symmetry breaking / M. Codish [et al.] // Constraints. — 2016. — Vol. 21, no. 3. — P. 375–393.