ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 519.17

ПОСТРОЕНИЕ ВСЕХ НЕИЗОМОРФНЫХ СУПЕРГРАФОВ БЕЗ ПРОВЕРКИ НА ИЗОМОРФИЗМ[1]

И. А. К. Камил, Х. Х. К. Судани, А. А. Лобов, М. Б. Абросимов

Саратовский национальный исследовательский государственный университет им. Н. Г. Чернышевского, г. Саратов, Россия

Важное направление в теории графов — построение графов с заданными свойствами без непосредственной проверки на изоморфизм. Программы, выполняющие такие построения, называются генераторами. Известны генераторы неориентированных графов с заданным числом вершин, деревьев, регулярных графов, двудольных графов, турниров и т. д. Граф $G = (V, \alpha)$ является частью графа $H = (W, \beta)$, если все вершины и рёбра графа G принадлежат H, то есть $V \subseteq W$ и $\alpha \subseteq \beta$. В этом случае граф H является суперграфом графа G. В исследованиях по теории графов часто свойства графа изучаются через какие-то его части. Возникает и обратная задача: по известной части построить графы, которые её содержат. Такая задача имеет место, например, при исследовании отказоустойчивых реализаций дискретных систем. В работе предлагается алгоритм построения для заданного графа всех его суперграфов с заданным числом рёбер без проверки на изоморфизм. Предлагается специальный матричный код и алгоритм генерации суперграфов методом канонических представителей на его основе. Рассматриваются оптимизации метода и вопросы, связанные с его параллельной реализацией.

Ключевые слова: *суперграф, изоморфизм, канонический код, исключение изоморфизма.*

DOI 10.17223/20710410/48/7

CONSTRUCTING ALL NONISOMORPHIC SUPERGRAPHS WITH ISOMORPHISM REJECTION

I. A. K. Kamil, H. H. K. Sudani, A. A. Lobov, M. B. Abrosimov

Saratov State University, Saratov, Russia

E-mail: kamil.iehab@mail.ru, hayder.1977@mail.ru, aisanekai@mail.ru, mic@rambler.ru

An important trend in graph theory is the construction of graphs with given properties without directly checking for isomorphism. Programs that perform such constructions are called generators. Generators of undirected graphs with a given number of vertices, trees, regular graphs, bipartite graphs, tournaments, etc. are well known. A graph $G = (V, \alpha)$ is a subgraph of a graph $H = (W, \beta)$ if all vertices and edges of G

[1]Работа выполнена при поддержке Минобрнауки России в рамках выполнения государственного задания (проект № FSRR-2020-0006).

belong to H, that is, $V \subseteq W$ and $\alpha \subseteq \beta$. If G is a subgraph of H, then H is a supergraph of G. In researches on graph theory, often the properties of a graph are studied through some of its parts. The inverse problem also appears: to construct graphs with given part as subgraph. Such a problem occurs, for example, in the study of fault-tolerant design of discrete systems. The algorithm for constructing for a given graph all its supergraphs with a given number of edges without checking for isomorphism is described. A special matrix code (route or M-code) and an algorithm for generating supergraphs by the method of canonical representatives based on it are proposed. The concept of the method of canonical representatives is that one representative in each class of isomorphic graphs is selected and is called canonical. A canonical representative function (canonical form) is a function c with the properties that $c(G) = c(H)$ if and only if G and H are isomorphic. The graph $c(G)$ is called the canonical representative of the graph G. We introduce a routing code (M-code) for the graph H relative to the graph G and denote $C_G(H)$. Given the adjacency matrices of the graphs G and H, construct the code $C_G(H)$ in two steps: first, write out the elements of the adjacency matrix of the graph H at the corresponding positions of which there is 1 in the adjacency matrix of the graph G, then those with a value of 0. For undirected graphs, only elements located above the main diagonal are written out. Optimization of the proposed method and issues related to its parallel implementation are discussed.

Keywords: *supergraph, isomorphism, canonical form, isomorphism rejection.*

1. Постановка задачи

Граф $G = (V, \alpha)$ является *частью* [1] (иногда используется термин «*подграф*» [2]) графа $H = (W, \beta)$, если все вершины и рёбра графа G принадлежат H, то есть $V \subseteq W$ и $\alpha \subseteq \beta$. В этом случае граф H является *суперграфом* (или *надграфом*) графа G.

Графы $G = (V, \alpha)$ и $H = (W, \beta)$ называют изоморфными и пишут $G \cong H$, если существует взаимно однозначное отображение $\varphi : V \to W$, такое, что

$$\forall u, v \in V \ \big((u, v) \in \alpha \Leftrightarrow (\varphi(u), \varphi(v)) \in \beta\big).$$

Граф $G = (V, \alpha)$ *вкладывается* в граф $H = (W, \beta)$, если существует инъективное отображение $\varphi : V \to W$, такое, что $\forall u, v \in V \ \big((u, v) \in \alpha \Rightarrow (\varphi(u), \varphi(v)) \in \beta\big)$. Обозначим вложение графов как $G \subseteq H$. Очевидно, что если граф G является частью графа H, то граф G вкладывается в граф H, однако обратное в общем случае неверно: если граф G вкладывается в граф H, то это означает, что в графе H есть часть, изоморфная графу G. Таким образом, если рассматривать графы с точностью до изоморфизма, то понятия «быть частью» и «вкладываться» будут синонимами.

Рассмотрим следующую задачу:

Задача 1. Для заданного n-вершинного графа G с m рёбрами необходимо построить все неизоморфные n-вершинные графы с $(m + k)$ рёбрами, в которые граф G вкладывается.

Или аналогичная формулировка:

Задача 2. Для заданного n-вершинного графа G с m рёбрами необходимо построить все его неизоморфные n-вершинные суперграфы с $(m + k)$ рёбрами.

Будем рассматривать неориентированные графы, хотя все методы могут быть непосредственно перенесены и на ориентированные графы. Через $n(G)$ и $m(G)$ будем обозначать соответственно число вершин и число рёбер графа G. Определения по теории графов, которые не даются в работе, можно найти в [1, 2].

Далее полагаем, без потери общности, что у рассматриваемых графов одинаковое множество вершин $V = \{v_1, \ldots, v_n\}$. Заметим, что одинаковое число вершин у графа G и его суперграфов не является существенным ограничением. Если нужно найти $(n+p)$-вершинные суперграфы n-вершинного графа G, то к графу G необходимо просто добавить p изолированных вершин.

В работе представлены алгоритмы, позволяющие решать поставленную задачу, и обоснования их корректности. Рассмотрены также вопросы, связанные с параллельной реализацией этих алгоритмов.

2. Алгоритмы с проверкой на изоморфизм

Простейшим способом решения задачи является алгоритм с проверкой на изоморфизм (алгоритм 1): перебираем все n-вершинные графы с заданным числом рёбер и каждый из них проверяем на вложение в него заданного графа. Если граф подходит и ранее не был найден изоморфный ему, то добавляем его в результирующий список графов.

Алгоритм 1. Построение суперграфов с непосредственной проверкой на изоморфизм

Вход: граф $G = (V, \alpha)$ с m рёбрами; целое $k \geqslant 0$.

Выход: R — множество всех попарно неизоморфных n-вершинных графов с $(m+k)$ рёбрами, в которые вкладывается граф G.

1: $R := \varnothing$.
2: **Для всех** $H = (V, \beta)$, таких, что $m(H) = m + k$:
3: **Если** $(G \subseteq H)\ \&\ \forall Q \in R\ (\neg(H \cong Q))$, **то**
$R := R \cup \{H\}$.
4: **Результат:** R.

Существует несколько способов улучшить алгоритм 1.

С одной стороны, многократную проверку на изоморфизм можно заменить однократным вычислением полного инварианта графа. Примером полного инварианта является максимальный матричный код. Матричный код представляет собой выписанные по строкам или столбцам элементы матрицы смежности. Получим некоторое число, если интерпретировать выписанную последовательность как запись числа в двоичной системе счисления. Матричный код не является инвариантом графа, так как изоморфные графы могут иметь отличающиеся матрицы смежности и соответственно матричные коды. Максимальный матричный код — максимальный из матричных кодов среди графов, изоморфных данному.

В настоящее время не известно ни эффективных алгоритмов для проверки изоморфизма графов, ни эффективных алгоритмов вычисления полного инварианта. Однако очередного кандидата нужно проверять на изоморфизм с каждым графом из результирующего списка, а полный инвариант нужно вычислить один раз, а затем сравнивать между собой числа. Известны методы быстрой генерации графов разного вида без проверки на изоморфизм. Хороший обзор можно найти в работе [3]. Например, в пакет nauty [4] входит программа geng, которая позволяет строить неориентированные графы с заданным числом вершин и некоторыми дополнительными ограничениями: с заданным числом рёбер, с заданными максимальной или минимальной степенями вершин. Так, если нас интересуют гамильтоновы 10-вершинные графы с 15 рёбрами, то с помощью утилиты geng можно построить все 10-вершинные двусвязные графы с 15 рёбрами, а затем проверить их на гамильтоновость собственной программой.

Такой подход используется, например, в [5]. Однако проверка на вложение является вычислительно сложной задачей, поэтому рассмотрим другой подход, который позволит такую проверку исключить. Действительно, достаточно перебирать те графы, частью которых является входной граф $G = (V, \alpha)$. Все эти графы можно получить добавлением рёбер к графу G. В этом случае проверять на вложение перебираемые графы не нужно, так как G является частью каждого из них. Обозначим через $\mathfrak{I}(G)$ функцию вычисления полного инварианта графа G; например, можно считать, что $\mathfrak{I}(G)$ вычисляет максимальный матричный код. Применив описанные улучшения, получим алгоритм 2.

Алгоритм 2. Построение суперграфов с проверкой на изоморфизм вычислением полного инварианта

Вход: граф $G = (V, \alpha)$ с m рёбрами; целое $k \geqslant 0$.

Выход: R — множество всех попарно неизоморфных n-вершинных графов с $(m + k)$ рёбрами, в которые вкладывается граф G.

1: $R := \varnothing$; $I := \varnothing$.
2: **Для всех** $H = (V, \beta)$, таких, что $\alpha \subseteq \beta$ & $m(H) = m + k$:
3: **Если** $\mathfrak{I}(H) \notin I$, **то**
$R := R \cup \{H\}$, $I := I \cup \{\mathfrak{I}(H)\}$.
4: **Результат:** R.

Как видно, в алгоритме 2 нужно хранить все найденные полные инварианты. Множество самих графов R можно не хранить, а восстановить графы при необходимости по множеству их полных инвариантов I, если полный инвариант это позволяет.

Рассмотрим способ генерации всех графов с $m + k$ рёбрами и заданной частью G с m рёбрами. Пусть $e_1, \ldots, e_t$ — рёбра, которые нужно добавить в граф G для получения полного графа. Для того чтобы получить все супеграфы, необходимо перебрать все возможные сочетания k рёбер из t возможных — это C_t^k вариантов. В алгоритме 1 необходимо перебирать $C_{n(n-1)/2}^{m+k}$ элементов, что больше, чем в алгоритме 2: $C_{n(n-1)/2-m}^{k}$. Можно разбить такой перебор на N непересекающихся частей следующим образом: выбираем целое $p < k$. Строим сочетания по p элементов из t и распределяем эти сочетания на N процессов. Каждый процесс перебирает варианты сочетаний с заданной заполненной частью и обрабатывает соответствующие ей графы. В полученных множествах могут содержаться изоморфные графы, поэтому далее необходимо их объединить и оставить по одному представителю от каждого класса изоморфных графов. В случае алгоритма 2 достаточно просто провести сравнение полных инвариантов.

Как видно, параллельные процессы в алгоритмах 1 и 2 не могут работать независимо друг от друга: для формирования конечного результата и его обработки нужно получить промежуточные результаты каждой части отдельно. Кроме того, необходимо хранить все построенные неизоморфные графы или их инварианты. От данных недостатков позволяет избавиться метод канонических представителей.

3. Метод канонических представителей

Идея метода канонических представителей состоит в том, что в каждом классе изоморфных графов выбирается один представитель, который называется каноническим [3]. Формально, вводится функция канонического представителя c, которая каждому графу G сопоставляет изоморфный ему граф $c(G)$ так, что $c(G) = c(H)$ тогда

и только тогда, когда G и H изоморфны. Граф $c(G)$ называется каноническим представителем графа G. Если рассматривать инвариант на основе матричного кода, то каноническим представителем обычно выбирается граф с минимальным или максимальным матричным кодом среди всех изоморфных ему (в случае неориентированных графов используются лишь элементы выше главной диагонали). Далее перебираются все подходящие графы и из них оставляются только канонические графы, множество которых и является ответом. В общем виде метод можно описать следующим образом [3]:

1) определяется способ кодирования графов;
2) среди всех кодов изоморфных графов выбирается канонический код (представитель);
3) порождаются все возможные коды графов;
4) порождённый граф принимается, если его код канонический, в противном случае исключается.

С использованием метода канонических представителей алгоритм 1 можно преобразовать в алгоритм 3. Удобно использовать функцию $C(G)$ проверки каноничности графа G, которая возвращает значение «истина», если $G = c(G)$, и «ложь» иначе, так как вычисление канонического представителя — более сложная задача, чем установление того, что заданный граф не является каноническим. Как видим, алгоритмы 1 и 3 отличаются лишь проверкой на изоморфизм и проверкой на каноничность.

Заметим, что в данном случае при распараллеливании каждый процесс действует независимо от другого, и так как в каждом классе изоморфизма каноничен лишь один граф, то он будет обработан только одним процессом. Это даёт возможность обрабатывать выдаваемые алгоритмом графы в потоковом режиме (построили граф, обработали, перешли к построению следующего) и не хранить их в памяти.

Алгоритм 3. Построение суперграфов методом канонических представителей

Вход: граф $G = (V, \alpha)$ с m рёбрами; целое $k \geqslant 0$.
Выход: R — множество всех попарно неизоморфных n-вершинных графов с $(m + k)$ рёбрами, в которые вкладывается G.

1: $R := \varnothing$.
2: **Для всех** $H = (V, \beta)$, таких, что $m(H) = m + k$:
3: **Если** $G \subseteq H$ & $C(H)$, **то**
$R := R \cup \{H\}$.
4: **Результат:** R.

Для того чтобы уменьшить перебор графов и исключить проверку вложения, как это сделано в алгоритме 2, необходимо, чтобы канонический представитель каждого класса изоморфизма, в который вкладывается граф G, присутствовал при переборе. Обычные максимальный или минимальный матричные коды графов для этого не подходят.

Рассмотрим пример. Требуется построить все суперграфы с одним дополнительным ребром для 5-вершинного цикла C_5. Для определённости будем считать каноническими представителями графы с максимальным матричным кодом. Так как мы рассматриваем неориентированные графы, в коде достаточно рассматривать только элементы, расположенные выше или ниже главной диагонали. Код будем составлять выписыванием элементов выше главной диагонали по столбцам слева направо или,

что то же самое, выписыванием элементов ниже главной диагонали по строкам сверху вниз. Матрица смежности цикла C_5 в канонической форме представлена на рис. 1,*а*. Заметим, что вопрос выбора формы представления исходного графа является нетривиальным и требует отдельного анализа. При добавлении одного ребра к циклу C_5 можно получить только один с точностью до изоморфизма граф. Например, добавим ребро между вершинами 1 и 4. Матрица смежности этого графа показана на рис. 1,*б*. Матрица смежности канонического представителя (то есть графа с максимальным матричным кодом) приведена на рис. 1,*в*. Жирным шрифтом выделены элементы, расположенные выше главной диагонали, которые участвуют в формировании матричного кода.

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

а

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

б

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

в

Рис. 1. Матрицы смежности графа C_5, его суперграфа с одним дополнительным ребром и изоморфного ему графа с максимальным матричным кодом

Как видно, этого представителя нельзя получить добавлением ребра к циклу C_5 в канонической форме. Аналогичный пример можно привести и для минимальных матричных кодов. Таким образом, обычные максимальные или минимальные матричные коды не позволяют доработать алгоритм 2 с учётом метода канонических представителей, поэтому необходимо сконструировать особый полный инвариант.

4. Маршрутный код

Маршрутный код (М-код) будем строить для графа H относительно графа G и обозначать $C_G(H)$. Пусть даны матрицы смежности графов G и H, построим код $C_G(H)$ следующим образом: сначала выписываем те элементы матрицы смежности графа H, на соответствующих позициях которых в матрице смежности графа G стоит 1, затем те, где стоит 0. Выписывать элементы можно по строкам или столбцам: в зависимости от задачи может быть предпочтительнее тот или другой способ. Для неориентированных графов выписываются только элементы, расположенные выше или ниже главной диагонали. Длина кода равна $n(n-1)/2$ бит. Для орграфов выписываются все элементы матрицы смежности и длина кода равна n^2 бит. Для определённости будем выписывать элементы матрицы смежности графа H выше главной диагонали по столбцам слева направо, как это показано на рис. 2. В качестве графов G и H возьмём 4-вершинные графы с двумя рёбрами. Их матрицы смежности $M(G)$ и $M(H)$ представлены на рис. 2; рядом в таблице показан порядок выписывания элементов матрицы $M(H)$ в коде C_G.

Код $C_G(G)$ графа G состоит из $m(G)$ начальных единиц, за которыми следуют нули. Если граф G с m рёбрами является частью графа H, то $C_G(H) \geqslant C_G(G)$, иначе $C_G(H) < C_G(G)$: если G является частью H, то $C_G(H)$ начинается не менее чем с m единиц, то есть он не меньше, чем код G. Если G не является частью H, то в первых

Рис. 2. Матрицы смежности графов G и H, порядок выписывания элементов кода C_G

m элементах кода графа H будет хотя бы один нуль, а значит, код графа H будет меньше кода любого графа, частью которого является G.

Если граф G вкладывается в граф H, то существует изоморфный H граф W, частью которого G является. Если G не является частью H, то $C_G(W) \geqslant C_G(G) \geqslant C_G(H)$, а это означает, что H не является максимальным. То есть среди графов с вложением G максимальным может быть только граф, частью которого является граф G, поэтому его можно получить добавлением рёбер к графу G. Будем называть максимальный из таких кодов *каноническим*, а соответствующий суперграф — *каноническим суперграфом*. Очевидно, что М-код является полным инвариантом графа, причём по этому коду естественным образом восстанавливается канонический представитель.

Алгоритм 4 строит все неизоморфные графы с вложением графа G, используя описанный полный инвариант.

Алгоритм 4. Построение суперграфов на основе М-кода

Вход: граф $G = (V, \alpha)$ с m рёбрами; целое $k \geqslant 0$.

Выход: R — множество всех попарно неизоморфных n-вершинных графов с $(m + k)$ рёбрами, в которые вкладывается G.

1: $R := \varnothing$, $I := \varnothing$.
2: **Для всех** $H = (V, \beta)$, таких, что $\alpha \subseteq \beta$ & $m(H) = m + k\}$:
3: **Если** $C_G(H) \notin I$ **то**
$R := R \cup \{H\}$.
4: **Результат:** R.

На практике метод канонических представителей редко используется в явном виде, обычно он оптимизируется для исключения представителей, которые не могут быть каноническими. Рассмотрим подход, который называют методом Рида — Фараджева [3]. Отметим следующее свойство максимального матричного кода (оно справедливо не только для М-кода).

Лемма 1. Если $c_1 = a_1 a_2 \ldots a_k \underline{1} 0 \ldots 0$ является каноническим матричным кодом, то $c_2 = a_1 a_2 \ldots a_k \underline{0} 0 \ldots 0$ также является каноническим матричным кодом.

Доказательство. Докажем методом от противного: пусть c_2 не максимален, H — соответствующий этому коду граф. Тогда существует граф, изоморфный H, такой, что его код $b = b_1 \ldots b_k b_{k+1} \ldots b_n$ больше, чем c_2. Это означает, что существует позиция i, для которой $a_i < b_i$ и $a_j = b_j$ при $j < i$. По определению кода количество единиц в c_2 и b одинаково (они получены перестановкой), а это значит, что они могут различаться только в позиции $i < k$. Применим тот же изоморфизм к графу с кодом c_1 и получим

изоморфный ему граф с кодом d. Коды d и b совпадают во всех позициях, кроме одной — t: в коде d там стоит 1, в коде b — 0. Если $t \geqslant i$, то d отличается от c_2 в позиции i (так как первые k элементов c_2 равны первым k элементам c_1, а $i < k$). Если $t < i$, то $d > c_1$, и отличаются они в элементах на позиции t. Так как найден код, больший c_1, получили противоречие, а значит, код c_2 — максимальный. ■

Таким образом, если заменить последнюю единицу на нуль в максимальном матричном коде, то снова получим максимальный матричный код, а значит, если код не максимальный, то, заменив любой нуль после последней единицы на единицу (добавив тем самым ребро в граф), нельзя получить максимальный матричный код. Это позволяет организовать отсечение кодов, которые не могут быть каноническими.

Составим дерево кодов n-вершинных графов следующим образом:

1) вершинами дерева являются все двоичные коды длины $l = n(n-1)/2$;
2) корнем является код $0 \ldots 0$;
3) у кода $0 \ldots 0$ потомками являются l кодов: $10 \ldots 00$, $01 \ldots 00$, $\ldots$, $00 \ldots 10$, $00 \ldots 01$;
4) у кода $a_1 \ldots a_k\underline{\mathbf{1}}00 \ldots 00$ потомками являются $(l-k-1)$ кодов: $a_1 \ldots a_k\underline{\mathbf{1}}10 \ldots 00$, $a_1 \ldots a_k\underline{\mathbf{1}}01 \ldots 00$, $\ldots$, $a_1 \ldots a_k\underline{\mathbf{1}}00 \ldots 10$ и $a_1 \ldots a_k\underline{\mathbf{1}}00 \ldots 01$.

В дереве кодов, согласно вышесказанному, можно выделить поддерево максимальных кодов, причём для графа G с кодом вида $1 \ldots 10 \ldots 0$ все коды графов, частью которых G является, будут его потомками. По лемме 1 предком канонического кода может быть только канонический код. Это означает, что среди потомков кода, не являющегося каноническим, канонических кодов не будет, и можно их не рассматривать. При использовании метода Рида — Фараджева происходит обход дерева кодов и проверяется каждая его вершина на каноничность: потомки вершины рассматриваются, только если код в ней канонический.

Канонический M-код также является максимальным матричным кодом — в нём изменён только порядок выписывания элементов, поэтому всё перечисленное верно и для него. Если дерево кодов основано на функции C_G, то в поддереве канонических кодов потомки кода графа G на расстоянии k являются всеми каноническими кодами суперграфов графа G с $m(G) + k$ рёбрами. На рис. 3 чёрные вершины соответствуют каноническим кодам $C_G(G)$, где в качестве графа G выбран граф, матрица смежности которого $M(G)$ приведена на рис. 2. Код $C_G(G) = 110000$ самого графа G показан большой вершиной. Для поиска суперграфов можно сразу выбрать в качестве корня дерева код $C_G(G)$. Этот метод описывается в алгоритме 5, который использует вспомогательную рекурсивную процедуру F (алгоритм 6).

Алгоритм 5. Построение суперграфов методом Рида — Фараджева

Вход: граф $G = (V, \alpha)$ с m рёбрами; целое $k \geqslant 0$.

Выход: R — множество всех попарно неизоморфных n-вершинных графов с $(m + k)$ рёбрами, в которые вкладывается G.

1: **Результат:** $F(\{G\}, k)$

Для уменьшения проверок каноничности можно использовать автоморфизмы графа. Напомним, что автоморфизмом графа $G = (V, \alpha)$ называется его изоморфизм на себя, то есть отображение $\varphi : V \to V$, такое, что $(u, v) \in \alpha \Leftrightarrow (\varphi(u), \varphi(v)) \in \alpha$. Пусть дан автоморфизм, переводящий вершину u_1 в v_1, u_2 в v_2 и между вершинами u_1 и u_2 нет ребра. Между вершинами v_1 и v_2 также нет ребра по определению автоморфизма.

Алгоритм 6. Процедура $F(\{G_1, \ldots, G_N\}, k)$

1: Обозначения: c_G — функция проверки каноничности на основе графа G; $\mathfrak{N}(H)$ — графы, соответствующие дочерним кодам для кода графа H в дереве кодов.
2: $S := \varnothing$.
3: **Если** $k = 0$, **то**
4: **Для всех** $i \in \{1, \ldots, N\}$:
5: **Если** $c_G(G_i)$, **то**
$S := S \cup G_i$;
6: **иначе**
7: **Для всех** $i \in \{1, \ldots, N\}$
$S := S \cup F(\mathfrak{N}(G_i), k-1)$.
8: **Вернуть** S.

Пусть граф G_u получен из G добавлением ребра (u_1, u_2), а граф G_v — добавлением ребра (v_1, v_2), тогда G_u и G_v изоморфны. Автоморфизм φ графа G является изоморфизмом G_u в G_v, так как рёбра исходного графа G переводятся друг в друга, а добавленное ребро G_u переходит в дополнительное ребро графа G_v. Коды таких графов отличаются лишь в одной позиции, и не составляет труда вычислить больший из них. Выявить подобные пары можно, перебрав все автоморфизмы графа G, что обычно выполняется полным перебором всех отображений множества вершин на себя. Проверка каноничности выполняется аналогичным перебором. Алгоритм 7 решает задачу проверки каноничности, пытаясь найти граф с бóльшим кодом. Если таких графов нет, то есть код является каноническим, то будут найдены и все возможные автоморфизмы графа, поданного на вход (очевидно, что два графа с равными кодами изоморфны). Автоморфизмы можно обрабатывать при их нахождении и не сохранять. В алгоритме используется рекурсивная процедура FA (алгоритм 8).

Алгоритм 7. Проверка графа на каноничность

Вход: $G = (V = \{v_1, \ldots v_n\}, \alpha)$; $H = (V, \beta)$, $\alpha \subseteq \beta$; целое $k \geqslant 0$.
Выход: Утверждение «Граф не каноничен», либо утверждение «Граф каноничен» и $\mathrm{Aut}(G)$ — набор всех автоморфизмов графа G.

1: $\mathrm{Aut}(G) := \varnothing$.
2: **Если** $FA(G, \varnothing, 0) =$ «Граф не каноничен», **то**
3: **Вернуть** «Граф не каноничен»;
4: **иначе**
5: **Вернуть** «Граф каноничен», $\mathrm{Aut}(G)$.

Алгоритм 8. Процедура $FA(G = (V, \alpha), \varphi = \{v_1 \to u_1, \ldots, v_i \to u_i\}, g)$

1: **Если** $|V| = |\varphi|$, **то**
2: **Если** $\neg g$, **то**
3: $\mathrm{Aut}(G) = \mathrm{Aut}(G) \cup \varphi$,
4: **Вернуть** «Продолжить поиск»;
5: **иначе**
6: **Вернуть** «Граф не каноничен»;
7: **иначе**
8: $U := V \setminus \{u_1, \ldots, u_i\}$.

9: **Для всех** $x \in U$
10: $p := \forall j \in \{1, \ldots, i\} \big((x, v_j) \in \alpha \Rightarrow (\varphi(x), \varphi(v_j)) \in \beta\big)$;
11: $q := \exists k \in \{1, \ldots, i\} \forall j \in \{1, \ldots, k-1\} \big(((x, v_j) \in \beta = (\varphi(x), \varphi(v_j)) \in \beta)$ & & $((x, v_k) \in \beta \neq (\varphi(x), \varphi(v_k)) \in \beta)\big)$.
12: **Если** p, **то**
13: **Если** $(q$ & $(\varphi(x), \varphi(v_k)) \in \beta) \vee g$, **то**
14: $t := FA(G, \{v_1 \to u_1, \ldots, v_i \to u_i, v_{i+1} \to x\}, 1)$.
15: **Если** $t =$ «Граф не каноничен», **то**
16: **Вернуть** «Граф не каноничен».
17: **Если** $\neg q$ & $\neg g$, **то**
18: $t := FA(G, \{v_1 \to u_1, \ldots, v_i \to u_i, v_{i+1} \to x\}, 0)$.
19: **Если** $t =$ «Граф не каноничен», **то**
20: **Вернуть** «Граф не каноничен».
21: **Вернуть** «Продолжить поиск».

Заключение

Рассмотрены несколько подходов к задаче построения для данного графа всех его суперграфов с заданным числом рёбер. С практической точки зрения наиболее эффективны методы построения графов без непосредственной проверки на изоморфизм. Алгоритм 5 разработан с использованием метода канонических представителей в форме Рида — Фараджева. Задача построения суперграфов часто возникает на практике; появление данной работы инициировано задачей построения всех гамильтоновых графов с заданным числом вершин и рёбер для анализа эффективности различных достаточных условий гамильтоновости [5]. Несмотря на то, что во многих случаях алгоритм показывает хорошие результы, иногда возникают ситуации, когда оказывается более предпочтительно использовать специальные генераторы. Например, для поиска регулярных гамильтоновых графов удобнее использовать генератор genreg [6], а для поиска кубических гамильтоновых графов — генератор snarkhunter [7].

Другая мотивация связана с задачами построения минимальных вершинных и рёберных k-расширений графов для анализа отказоустойчивости дискретных систем [8]. Основные идеи решения задачи о построении всех неизоморфных суперграфов с заданным числом рёбер, а также описанный в работе маршрутный код появились именно при решении задачи построения вершинных и рёберных k-расширений в работах [9, 10]. Алгоритм 5 реализован на языке C++ с использованием MPI и протестирован на кластере Поволжского регионального центра новых информационных технологий (ПРЦ НИТ) [11]. Некоторые вопросы остались за пределами рассмотрения в данной работе. Например, форма представления исходного графа существенным образом влияет на скорость генерации его суперграфов. Экспериментальные вычисления позволяют сделать вывод, что в среднем хорошо себя показывает выбор графа в форме максимального матричного кода, однако этот вопрос требует дополнительного анализа.

ЛИТЕРАТУРА

1. *Богомолов А. М., Салий В. Н.* Алгебраические основы теории дискретных систем. М.: Наука, 1997. 368 с.
2. *Харари Ф.* Теория графов. М.: Мир, 1973. 300 с.
3. *Brinkmann G.* Isomorphism rejection in structure generation programs // DIMACS Ser. Discr. Math. Theor. Comput. Sci. 2000. V. 51. P. 25–38.

4. *McKay B. D. and Piperno A.* Practical graph isomorphism. II // J. Symbolic Computation. 2014. No. 60. P. 94–112.
5. *Абросимов М. Б.* Сравнение достаточных условий гамильтоновости графа, основанных на степенях вершин // Прикладная дискретная математика. 2019. № 45. С. 55–63.
6. *Абросимов М. Б.* Графовые модели отказоустойчивости. Саратов: Изд-во Сарат. ун-та, 2012. 192 с.
7. *Meringer M.* Fast generation of regular graphs and construction of cages // J. Graph Theory. 1999. V. 30. P. 137–146.
8. *Brinkmann G., Goedgebeur J., and McKay B. D.* Generation of cubic graphs // Discr. Math. Theoret. Comput. Sci. 2011. V. 13(2). P. 69–80.
9. *Абросимов М. Б., Камил И. А. К., Лобов А. А.* Построение всех неизоморфных минимальных вершинных расширений графа методом канонических представителей // Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика. 2019. Т. 19. Вып. 4. С. 479–485.
10. *Абросимов М. Б., Судани Х. Х. К., Лобов А. А.* Построение минимальных рёберных расширений графа без проверки на изоморфизм // Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика. 2020. Т. 20. Вып. 1. С. 105–115.
11. `http://prcnit.sgu.ru` — Официальный сайт Поволжского регионального центра новых информационных технологий, 2020.

REFERENCES

1. *Bogomolov A. M. and Salii V. N.* Algebraicheskie osnovy teorii diskretnykh sistem [Algebraic Foundations of the Theory of Discrete Systems]. Moscow, Nauka Publ., 1997, 368 p. (in Russian)
2. *Harary F.* Graph Theory. Addison-Wesley, 1969. 274 p.
3. *Brinkmann G.* Isomorphism rejection in structure generation programs. DIMACS Ser. Discr. Math. Theor. Comput. Sci., 2000, vol. 51, pp. 25–38.
4. *McKay B. D. and Piperno A.* Practical graph isomorphism, II. J. Symbolic Computation, 2014, no. 60, pp. 94–112.
5. *Abrosimov M. B.* Sravneniye dostatochnykh usloviy gamil'tonovosti grafa, osnovannykh na stepenyakh vershin [Comparison of sufficient degree based conditions for Hamiltonian graph]. Prikladnaya Diskretnaya Matematika, 2019, no. 45. pp. 55–63. (in Russian)
6. *Meringer M.* Fast generation of regular graphs and construction of cages. J. Graph Theory, 1999, vol. 30, pp. 137–146.
7. *Brinkmann G., Goedgebeur J., and McKay B. D.* Generation of cubic graphs. Discr. Math. Theor. Comput. Sci., 2011, vol. 13(2), pp. 69–80.
8. *Abrosimov M. B.* Grafovye modeli otkazoustoichivosti [Fault Tolerance Graph Models]. Saratov, Saratov Univ. Publ., 2012, 192 p. (in Russian)
9. *Abrosimov M. B., Kamil I. A. K., and Lobov A. A.* Postroenie vsekh neizomorfnykh minimal'nykh vershinnykh rasshireniy grafa metodom kanonicheskikh predstaviteley [Construction of all nonisomorphic minimal vertex extensions of the graph by the method of canonical representatives]. Izv. Saratov Univ. (N. S.), Ser. Math. Mech. Inform., 2019, vol. 19, iss. 4, pp. 479–486. (in Russian)
10. *Abrosimov M. B., Sudani H. H. K., and Lobov A. A.* Postroenie minimal'nykh rebernykh rasshireniy grafa bez proverki na izomorfizm [Construction of all minimal edge extensions of the graph with isomorphism rejection]. Izv. Saratov Univ. (N. S.), Ser. Math. Mech. Inform., 2020, vol. 20, iss. 1, pp. 105–115. (in Russian)
11. `http://prcnit.sgu.ru` — Volga Regional Center for New Information Technologies, 2020.