Секция 7

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 519.17

О ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ РАСШИРЕНИЙ ГРАФОВ

М. Б. Абросимов

В 1976 году Хейз в работе [1] предложил основанную на графах модель для исследования отказоустойчивости. Технической системе Σ сопоставляется помеченный граф G, вершины которого соответствуют элементам системы Σ, ребра (или дуги) — связям между элементами, а метки указывают тип элементов. Под отказом элемента системы Σ понимается удаление соответствующей вершины из графа системы G и всех связанных с ней ребер. Говорят, что система Σ^* является *k-отказоустойчивой реализацией* системы Σ, если отказ любых k элементов системы Σ^* приводит к графу, в который можно вложить граф системы Σ с учетом меток вершин. Построение k-отказоустойчивой реализации системы можно представить себе как введение в нее определенного числа новых элементов и связей. При этом предполагается, что в нормальном режиме работы избыточные элементы и связи *маскируются*, а в случае отказа происходит *реконфигурация* системы до исходной структуры.

Пусть в системе Σ встречается t различных типов элементов. Очевидно, что любая ее k-отказоустойчивая реализация должна содержать не менее k дополнительных элементов каждого типа. Легко видеть, что такого числа дополнительных элементов достаточно для построения k-отказоустойчивой реализации системы Σ. В самом деле, добавим k элементов каждого типа и соединим их все между собой и с элементами системы Σ. Тогда любой отказавший элемент можно будет заменить одним из добавленных элементов соответствующего типа.

k-отказоустойчивая реализация Σ^* системы Σ, состоящей из элементов t различных типов, называется *оптимальной*, если система Σ^* отличается от системы Σ на k элементов каждого из t типов системы Σ, и среди всех k-отказоустойчивых реализаций с тем же числом элементов система Σ^* имеет наименьшее число связей.

Хейз предложил процедуры построения оптимальной k-отказоустойчивой реализации для цепи, цикла и помеченного дерева. Позднее Хейз и Харари в работе [2] обобщили модель на случай отказов связей между элементами, предложив понятие *реберной отказоустойчивости*. Модель отказоустойчивости, в которой рассматриваются отказы элементов, в работе [3] было предложено называть *вершинной отказоустойчивостью*.

Если исключить понятие отказа, то рассмотренные ранее понятия могут быть естественным образом сформулированы с использованием обычных терминов теории графов. Впервые это было предложено автором в работе [4], и далее мы дадим формальные определения именно в таком виде.

Граф $G^* = (V^*, \alpha^*)$ называется *вершинным* (*реберным*) *k-расширением* (k — натуральное) графа $G = (V, \alpha)$, если граф G вкладывается в каждый подграф графа G^*, получающийся удалением любых его k вершин (ребер).

Сформулируем задачу о вершинном k-расширении как задачу распознавания свойств, то есть задачу, ответом на которую может быть «да» или «нет»:

ВЕРШИННОЕ k-РАСШИРЕНИЕ
УСЛОВИЕ. Даны графы $G = (V, \alpha)$ и $H = (U, \beta)$.
ВОПРОС. Верно ли, что граф G является вершинным k-расширением графа H?

Теорема 1. Задача ВЕРШИННОЕ k-РАСШИРЕНИЕ является NP-полной.

Аналогично формулируется задача о реберном k-расширении:

РЕБЕРНОЕ k-РАСШИРЕНИЕ
УСЛОВИЕ. Даны графы $G = (V, \alpha)$ и $H = (U, \beta)$.
ВОПРОС. Верно ли, что граф G является реберным k-расширением графа H?

Теорема 2. Задача РЕБЕРНОЕ k-РАСШИРЕНИЕ является NP-полной.

ЛИТЕРАТУРА

1. *Hayes J. P.* A graph model for fault-tolerant computing system // IEEE Trans. Comput. 1976. V. C.-25. No. 9. P. 875–884.
2. *Harary F., Hayes J. P.* Edge fault tolerance in graphs // Networks. 1993. V. 23. P. 135–142.
3. *Harary F., Hayes J. P.* Node fault tolerance in graphs // Networks. 1996. V. 27. P. 19–23.
4. *Абросимов М. Б.* Минимальные расширения графов // Новые информационные технологии в исследовании дискретных структур. Томск, 2008. С. 59–64.

УДК 519.171

ПОСТРОЕНИЕ ПОКРЫТИЙ РЁБЕР ГРАФА КЛИКАМИ

И. А. Бадеха, П. В. Ролдугин

Реберным покрытием кликами (РПК) графа G называется такой набор клик (полных подграфов) $K_1, ..., K_r$, что любое ребро графа G лежит хотя бы в одной из этих клик. Задача построения РПК, минимального по числу входящих в него клик, как известно, является NP-полной (см., например, [1]). Поскольку каждую из клик в наборе можно дополнить произвольным образом до максимальной клики и получить также РПК, то будем считать, что в определении РПК и далее речь идет о максимальных кликах.

Легко выделить клики, которые обязаны содержаться в каждом РПК для данного графа. Назовем ребро e графа G собственным ребром клики K, если оно лежит в этой клике и не лежит ни в какой другой клике графа G. Соответственно клику K, имеющую хотя бы одно собственное ребро, назовем зафиксированной.

Как известно, вершина графа называется *доминирующей*, если она соединена ребром с каждой из остальных вершин графа.

Определение 1. *Конструкцией* C в графе G будем называть порожденный подграф графа G, обладающий ненулевым количеством доминирующих вершин, и такой, что граф, получающийся из графа C удалением всех доминирующих вершин и инцидентных им рёбер, является непустым и связным.

В работе доказывается, что в связном графе без собственных рёбер существует конструкция.

Конструкция C может быть пригодной или непригодной для разбиения графа, что определяется свойствами рёбер, входящих в неё. Данные свойства подробно исследованы в работе. Все рёбра пригодной конструкции разбиваются на две категории: те,

которые в минимальном покрытии должны покрываться кликой, содержащейся внутри конструкции (категория 1), и те, которые могут покрываться вне клики без потери свойства минимальности (категория 2).

Основным содержанием работы можно считать следующее утверждение:

Теорема 1. Пусть $R_1, ..., R_s$ — минимальное РПК конструкции C, пригодной для разбиения графа, с вычеркнутыми рёбрами категории 2. Пусть также $L_1, ..., L_r$ — минимальное РПК оставшейся части графа G. Тогда $R_1, ..., R_s, L_1, ..., L_r$ — минимальное РПК графа G.

В работе приводится эвристический алгоритм поиска рёберных покрытий в графе, основанный на использовании данной теоремы и состоящий из следующих основных шагов:

1. Поиск собственных рёбер в графе. Фиксирование соответствующих клик.
2. Поиск конструкции, пригодной для разбиения графа. В случае, если пригодных конструкций в графе не существует, выбирается конструкция, наиболее приближенная к пригодной для разбиения. В случае, если ни одной конструкции не было найдено, для процедуры разбиения в качестве аналога разбивающей конструкции выбирается наименьшее по мощности окружение ребра.
3. Разбиение исходного графа на два графа: в первый из графов входят те рёбра, которые относятся к конструкции, а во второй граф — все остальные рёбра.
4. Рекурсивное применение данного алгоритма к обоим графам.
5. Объединение полученных покрытий.

Для некоторых классов графов данный алгоритм является точным. В работе представлены такие графы.

ЛИТЕРАТУРА

1. *Orlin J.* Contentment in graph theory: Covering graphs with cliques. // Indagationes Math. 1977. V. 39. P. 406–424.

УДК 519.5

РАСПОЗНАВАНИЕ ГРАФА ПРИ ПОМОЩИ БЛУЖДАЮЩЕГО ПО НЕМУ АГЕНТА

И. С. Грунский, Е. А. Татаринов

Основной проблемой компьютерной науки является проблема взаимодействия управляющей и управляемой систем (управляющего автомата, агента и его операционной среды) [1]. Взаимодействие этих систем зачастую представляется как процесс перемещения агента по помеченному графу (лабиринту) среды [2]. Определился ряд подходов к моделированию операционных сред, одним из которых является топологический [3]. В этом случае агенту недоступна метрическая или алгоритмическая информация о среде и доступна только информация о связях между различными областями среды.

Постановка задачи. Рассматривается конечный граф G — неориентированный, связный, без петель и кратных ребер, где V — множество его вершин и E — множество ребер. Вершины и инциденторы графа G можно метить специальными красками и/или камнями, где инцидентор — «точка прикосновения» вершины v и ребра (v, u). Изначально предполагается, что все вершины и инциденторы не помечены.

Требуется построить такое множество $C = \{c_i\}$, где $c_i = \{V_i, E_i\}$ и $V_i \subseteq V$, $E_i \subseteq E$, и найти такое отображение A, чтобы, используя их, можно было построить граф H, изоморфный графу G, и тем самым распознать граф G с точностью до изоморфизма. Под отображением будем понимать установление взаимно однозначного соответствия между вершинами и ребрами каждого конкретного c_i и вершинами и ребрами графа H. Для проведения проверок и выполнения отображения используется агент, который передвигается по неизвестному графу G из вершины в вершину по ребру, соединяющему их. Агент воспринимает некоторую локальную информацию о 1-окрестности v и использует краски и/или камни из множества $\{b, r, L\}$ для пометки вершин и инцидентов графа G. Он обладает конечной, но бесконечно наращиваемой памятью, в которую будут отображаться проверки и строиться таблица графа H .

В данной работе предполагается, что c_i — это M-пути и, возможно, одно обратное ребро. Обратное ребро — ребро, соединяющее две вершины M-пути. M-путь [4] — это простой путь в графе, в котором каждая следующая вершина имеет больший M-номер, чем предыдущая. M-нумерацией вершин графа называется нумерация, соответствующая порядку их обхода при поиске в глубину [4]. Правило построения проверок c_i следующее: отбирается M-путь, из конечной вершины которого можно попасть только в ранее пройденную вершину. Агент обладает красками r, b и камнем L. Краска r используется для пометки древесных ребер, а b — для пометки уже исследованных вершин и инцидентов. Камень L используется при исследовании обратного ребра. Агент, блуждая по графу, создает неявную M-нумерацию в графе G, которую отображает в граф H, создавая в нем явную M-нумерацию. Явная M-нумерация позволяет корректно отобразить проверки и обратные ребра. Агент обозревает метки 1-окрестности вершины, в которой он находится: метку на вершине и метку на ближних и дальних инцидентах ребер из 1-окрестности. Процесс выполнения алгоритма разбит на следующие этапы: 1) построение множества проверок; 2) выполнение проверки, построенной по правилу; 3) отображение конкретной проверки в граф H; 4) отображение в граф H обратного ребра из проверки, если таковое есть.

Разработаны полиномиальные алгоритмы «Покрытие» и «Отображение» для выполнения шагов 1–2 и 3–4 соответственно. Алгоритмы могут выполняться как последовательно, так и параллельно. В работе рассматривается параллельное их выполнение.

Теорема 1. Алгоритмы «Покрытие» и «Отображение» распознают граф G с точностью до изоморфизма. Они требуют две различные метки, их временная сложность ограничена снизу линейной функцией от числа вершин n графа G, а сверху — кубической функцией. Емкостная сложность равна $O(n^2)$.

Предложенный алгоритм охватывает более широкий класс графов, чем предложенный в работе [5], хотя верхняя оценка временной сложности на порядок хуже.

ЛИТЕРАТУРА

1. *Капитонова Ю. В., Летичевский А. А.* Математическая теория программирования вычислительных систем. М.: Наука, 1988. 296 с.
2. *Кудрявцев В. Б., Ущумлич Ш., Килибарда Г.* О поведении автоматов в лабиринтах // Дискретная математика. 1993. Т. 4. Вып. 3. С. 3–26.
3. *Kuipers B.* The spatial semantic hierarchy // Artifical Intellegence. 2000. V. 119. No. 1–2. P. 191—233.
4. *Евстигнеев В. А.* Применение теории графов в программировании. М.: Наука, 1985. 352 с.

5. *Грунский И. С., Татаринов Е. А.* Алгоритм распознавания графов // Труды Четвертой Междунар. конф. «Параллельные вычисления и задачи управления» PACO'2008. М.: Институт проблем управления им. В. А. Трапезникова РАН, 2008. С. 1483–1498.

УДК 519.17

К ВОПРОСУ О ТОЧНЫХ РАСШИРЕНИЯХ ТУРНИРОВ

А. А. Долгов

Ориентированным графом называется пара $G = (V, \alpha)$, где V — конечное непустое множество, называемое *множеством вершин*, а α — отношение на множестве вершин V, называемое *отношением смежности*.

Граф с антисимметричным отношением смежности называется *направленным графом*, или *диграфом*. Полный диграф без петель называется *турниром*.

Граф H называется *точным k-расширением* графа G, если G изоморфен каждому подграфу H, получающемуся путем удаления любых его k вершин.

Первое из известных семейств точных расширений турниров — семейство транзитивных турниров — было описано в работе [1]. *Транзитивный турнир* — это турнир, у которого из существования дуг (u, v) и (v, w) вытекает существование дуги (u, w). В работе показано, что точным k-расширением турнира T_n при $n > 2$ является транзитивный турнир T_{n+k}.

В работе [2] рассматривается схема построения семейства вершинно-симметрических турниров. Оказывается, что графы этого семейства обладают циклической симметрией.

Удалось показать, что любой циклически-симметричный граф является точным 1-расширением. Семейство, описанное в [2], не является единственным семейством турниров с циклической симметрией. В докладе представляется схема, позволяющая построить все графы с заданным числом вершин, обладающие циклической симметрией. Также указывается необходимая модификация алгоритма, позволяющая получить с его помощью только циклически-симметричные турниры.

Рассмотрим операцию над парой графов G_1 и G_2, назовем ее операцией вершинной подстановки графа G_1 в граф G_2.

Результатом вершинной подстановки графа $G_1 = (V_1, \alpha_1)$ в $G_2 = (V_2, \alpha_2)$, где $|V_1| = n_1$, $|V_2| = n_2$, будет граф $G = (V, \alpha)$, такой, что:

1) $V = \{v_{i,j} | i = \overline{1, n_1}, j = \overline{1, n_2}\}, |V| = n_1 \times n_2$;
2) $(v_{i,k}, v_{j,t}) \in \alpha$, если $k = t$ и $(v_i, v_j) \in \alpha_1$ или если $k \neq t$ и $(v_k, v_t) \in \alpha_2$.

Удалось доказать, что граф, получающийся в результате вершинной подстановки транзитивного турнира в циклически-симметричный турнир, является точным 1-расширением, отличным от графов, принадлежащих описанным семействам.

ЛИТЕРАТУРА

1. *Абросимов М. Б.* Минимальные расширения транзитивных турниров // Вестник Томского госуниверситета. Приложение. 2006. № 17. С. 187–190.
2. *Абросимов М. Б., Долгов А. А.* Точные расширения некоторых турниров // Вестник Томского госуниверситета. Приложение. 2007. № 23. С. 211–216.

УДК 519.19

АЛГОРИТМЫ ГЕНЕРАЦИИ КОРНЕВЫХ ДЕРЕВЬЕВ НА ОСНОВЕ ПРОЦЕДУРЫ ПОЛНОГО РАЗБИЕНИЯ

В. В. Кручинин

Деревья являются одним из важнейших классов комбинаторных множеств, которые достаточно хорошо изучены. Однако можно предложить ряд алгоритмов генерации классов корневых деревьев, основанных на процедуре полного разбиения. Впервые подобную процедуру предложил Н. Я. Виленкин как подсчет процессов последовательных разбиений [1]. Р. Стенли предложил использовать эту процедуру для полного разбиения конечного множества S, в результате выполнения которой получается некоторое корневое дерево [2]. Основная идея данного доклада заключается в том, что эту процедуру можно модифицировать, используя вместо конечного множества S натуральное число n. Тогда, имея некоторый алгоритм $R(n)$ представления натурального числа n в виде суммы, в общем случае, неотрицательных чисел λ_i, строится непомеченное n-узловое корневое дерево. Алгоритм построения дерева следующий.

1) Создаем корень дерева r и пару $< n, r >$ заносим в стек.

2) Если стек пуст, то завершаем процедуру.

3) Вынимаем из стека очередную пару $< k, z >$. Если $k = 1$, то данный узел становится листом, и переходим на шаг 2, иначе, в соответствии с алгоритмом R, получаем представление $\pi : [\lambda_1 + \lambda_2 + \cdots + \lambda_m]$, такое, что $\sum_{i=1}^{m} \lambda_i = k - 1$, и переходим на шаг 4.

4) Создаем m узлов $\{s_i\}_{i=1}^{m}$, присоединяем их в качестве сыновей к узлу z в соответствии с порядком, заданным в представлении $\{\lambda_i\}_{i=1}^{m}$. Все пары $< \lambda_i, s_i >$ записываем в стек. Переходим на шаг 3.

Рассмотрим свойства предложенного алгоритма полного разбиения.

1. Если задан $R(n)$ и $\lambda_i \geqslant 1$ для всех представлений $\pi \in P_{n-1}$, то предложенный алгоритм всегда остановится.

Это утверждение основано на условии, что в дереве с n узлами имеется корень, следовательно, число $n - 1$ является количеством узлов во всех поддеревьях сыновей корня. Алгоритм R получает некоторое представление $\lambda_1 + \lambda_2 + \cdots + \lambda_m = n - 1$, где λ_i определяет число узлов в поддереве i-го сына и $1 \leqslant \lambda_i \leqslant n - 1$. Тем самым, последовательно применяя алгоритм R к числам, меньшим чем n, получим останов. Поскольку алгоритм строит корневое дерево и всегда останавливается, то он всегда строит дерево.

2. Число деревьев для заданного числа узлов равно

$$K(n) = \sum_{\pi \in P_{n-1}} \prod_{i=1}^{k} K(\lambda_i), \tag{1}$$

где $\pi = \{\lambda_1 + \lambda_2 + \cdots + \lambda_k = n - 1\}$; P_{n-1} — множество представлений числа $n - 1$ в соответствии с алгоритмом R.

Применим этот алгоритм для генерации n-узловых корневых упорядоченных деревьев. Поскольку в таких деревьях порядок следования сыновей в узле важен и сумма чисел узлов в поддеревьях сыновей равна $n-1$, то для генерации таких деревьев можно использовать процедуру полного разбиения, основанную на генерации композиций

натурального числа n. Тогда

$$K(n+1) = \sum_{\pi \in S_n} \prod_{i=1}^{k} K(\lambda_i), \tag{2}$$

где $\pi = \{\lambda_1 + \lambda_2 + \cdots + \lambda_k = n\}$; S_n — множество композиций числа n. Известно также [2], что $K(n+1) = C_n$, где C_n — число Каталана. Тогда получим следующее тождество для чисел Каталана:

$$C_n = \sum_{\pi \in S_n} \prod_{i=1}^{k} C_{\lambda_i - 1}.$$

Из выражения (2) следует, что каждая композиция порождает некоторое подмножество корневых упорядоченных деревьев. Для построения эффективного генератора [3] необходимо знать число деревьев для каждой композиции $\pi \in S_n$. Обозначим через $T(i)$ число корневых упорядоченных деревьев для i-й композиции в лексикографическом упорядочении композиций. С использованием свойств композиций было получено выражение для функции $T(i)$:

$$T(i) = \begin{cases} 1, & i = 0, \\ T(i - 2^m)\frac{4l+2}{l+2}, & i > 0. \end{cases} \tag{3}$$

где $m = \lfloor \log_2(i) \rfloor$, $l = \begin{cases} m - \lfloor \log_2(2^{m+1} - i - 1) \rfloor - 1, & 0 \leqslant i < 2^{m+1} - 1, \\ m, & i = 2^{m+1} - 1. \end{cases}$

На рис. 1 изображен график функции $T(i)$. Свойства этой функции таковы:
1) $\sum_{i=0}^{2^m-1} T(i) = C_m$, где C_m — число Каталана.
2) $T(2^m - 1) = C_{m-1}$.
3) В точках $i = 2^m - 1$ функция $T(i)$ будет иметь локальные максимумы.

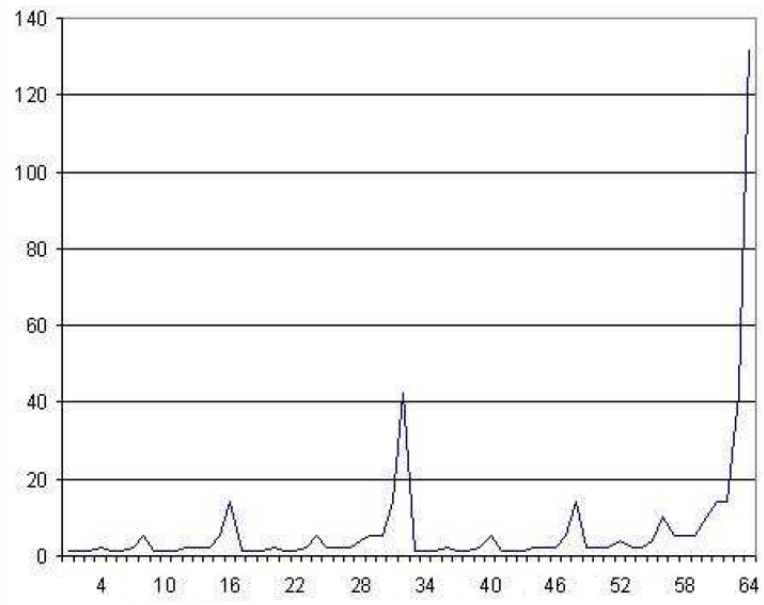

Рис. 1. График функции $T(i)$

В докладе будут подробно рассмотрены алгоритмы генерации корневых деревьев с использованием процедуры полного разбиения, основанной на разложениях, композициях и разбиениях натурального числа n.

ЛИТЕРАТУРА

1. *Виленкин Н. Я.* Комбинаторика. М.: Наука, 1969.
2. *Стенли Р.* Перечислительная комбинаторика. Деревья, производящие функции и симметрические функции. М.: Мир, 2005.
3. *Кручинин В. В.* Методы построения алгоритмов генерации и нумерации комбинаторных объектов на основе деревьев И/ИЛИ. Томск: Изд-во «В-Спектр», 2007.

УДК 519.175.1

АЛГОРИТМЫ ПРОВЕРКИ ИЗОМОРФИЗМА ГРАФОВ НА ОСНОВЕ ИХ ПОСЛЕДОВАТЕЛЬНОЙ СОГЛАСОВАННОЙ ДЕРЕГУЛЯРИЗАЦИИ

И. В. Широков, А. В. Пролубников

В задаче проверки изоморфизма графов (задача ИГ) даны два обыкновенных графа с одинаковым числом вершин и ребер. Необходимо ответить на вопрос, существует ли такое биективное отображение (изоморфизм) множества вершин одного графа на множество вершин второго, которое сохраняло бы смежность соответствующих вершин?

По причине неопределенности своего положения в иерархии теории сложности, задача ИГ имеет большое теоретическое значение, а также часто возникает и в приложениях. Алгоритмы решения задачи ИГ используются при решении многих прикладных задач: в задачах распознавания образов, в протоколах доказательства с нулевым разглашением; к задаче ИГ может быть сведена и вычислительно-эффективно решена задача дешифрования шифра двойной перестановки [1]. Поскольку для вычислительно сложных случаев задачи ИГ не разработано полиномиальных алгоритмов решения, возможно построение криптографических схем, основанных на вычислительной сложности решения для них задачи ИГ, например, как в [2].

Под последовательной согласованной дерегуляризацией пары графов мы понимаем такое последовательное изменение элементов матриц смежности графов (весов вершин и ребер), вследствие которого понижается мощность групп автоморфизмов графов при сохранении равенства некоторых вычисляемых в ходе дерегуляризации инвариантных характеристик графов. В докладе рассматриваются полиномиальные схемы алгоритмов для задачи ИГ, основанные на таком подходе и в качестве инварианта использующие элементы обратной матрицы к модифицированной матрице смежности.

Предложенные алгоритмы протестированы на библиотеке задач [3]. Не найдено примеров неправильного решения или невозможности нахождения решения представленными алгоритмами задачи ИГ для деревьев, случайных, планарных графов, регулярных k-мерных сеток ($k \leqslant 4$) и некоторых других классов графов. Установлено, что алгоритм решает и те задачи изоморфизма графов из библиотеки, на которых время работы алгоритмов Ullman и NAUTY — наиболее эффективных алгоритмов решения общего случая задачи ИГ — становится экспоненциальным при некоторой нумерации вершин [4]. Кроме этого, задача ИГ успешно решается предложенными алгоритмами и для сильно регулярных графов из наиболее обширной их библиотеки [5] — эти графы представляют собой класс, для которого задача проверки изоморфизма графов име-

ет наибольшую вычислительную сложность. Получены орбиты групп автоморфизмов всех графов, представленных в библиотеке [5], что также свидетельствует об эффективности предложенного подхода.

ЛИТЕРАТУРА

1. *Faizullin R. T., Prolubnikov A. V.* An algorithm of the spectral splitting for the double permutation cipher // Recognition and Image Analysis. MAIK, Nauka. 2002. V. 12. No. 4. P. 310–324.
2. *Пролубников А. В., Файзуллин Р. Т.* Построение защищенного видеоканала с использованием изоморфизма графов // Вестник Томского госуниверситета. Приложение. №9(I). 2004. С. 71–74.
3. *Foggia P., Sansone C., Vento M.* A Database of graphs for isomorphism and sub-graph isomorphism benchmarking // Proc. of the 3rd IAPR TC-15 international workshop on graph-based representations, Italy, 2001. P. 157–168.
4. *Miyazaki T.* The complexity of McKay's canonical labeling algorithm // Groups and Computation, II. Amer. Math. Soc., Providence, RI, 1997. P. 239–256.
5. `http://www.maths.gla.ac.uk/~es` — Strongly Regular Graphs.