Информационные процессы, Том 11, № 2, 2011, стр. 225–252.

МАТЕМАТИЧЕСКИЕ МОДЕЛИ, ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

Учёт обхвата при подсчёте коротких циклов в двудольных графах

А.Н.Воропаев

Петрозаводский государственный университет, Петрозаводск, Россия
Поступила в редколлегию 14.06.2011

Аннотация—Рассмотрены пять методов подсчёта циклов в двудольных графах с длиной, близкой к обхвату. Для явных формул показано, что наиболее трудоёмкой операцией оказывается умножение матриц, если длины подсчитываемых циклов не превышают удвоенного обхвата. Детально обсуждаются количество матриц и матричных умножений, участвующих в расчётах с использованием явных выражений и алгоритма «леденцов» (которые имеют одинаковые порядки сложности). Выполнено сравнение вычислительных затрат всех методов в случае подсчёта циклов длиной до удвоенного обхвата в графах различной плотности. В частности, указаны оценки плотности графов, при которой алгоритмы, ориентированные на разреженную структуру, работают эффективнее «матричных» методов.

1. ВВЕДЕНИЕ

Количество циклов различной длины является важной характеристикой графов многих систем — реальных сетей, решёточных моделей статистической механики, кодов с низкой плотностью проверок на чётность (LDPC). Для фиксированных значений $k \leqslant 13$ существуют явные формулы [1, 2, 3], выражающие количество циклов длиной k через матрицу смежности графа. Вычислительная сложность этих формул при $k \leqslant 7$ имеет тот же порядок, что умножение $n \times n$-матриц, а при $k \geqslant 8$ является величиной $O(n^{[k/2]} \log n)$, где n — количество вершин в графе. Ещё более универсальные методы подсчёта циклов, которые формально применимы для любых графов и любых значений длины цикла, имеют сложность $O(n^{k-1} \log n)$, $O(n^k \log n)$ или бóльшую, как показано в [4, 5]. Известно, что сложность подсчёта циклов длиной k в произвольных графах неизбежно увеличивается с ростом k [6].

Поскольку универсальные подходы к подсчёту циклов не слишком продуктивны, представляет интерес разработка специализированных алгоритмов для частных семейств графов, которые характеризовались бы меньшей скоростью роста сложности. Например, в [3] рассматривались как формулы для произвольных графов, так и аналогичные по структуре выражения для двудольных графов, сложность которых, по крайней мере, в случаях $k = 8, 10, 14$ оказывается на порядок меньше.

Ряд работ посвящён подсчёту коротких циклов в двудольных графах LDPC-кодов, для которых характерны небольшая плотность и значение обхвата $g \geqslant 6$. Особенность большинства рассматриваемых далее методов состоит в том, что за счёт ограничения длины цикла (относительно обхвата) достигается время работы $O(n^4)$. Так, авторы [7] разработали алгоритм нахождения обхвата g двудольного графа и подсчёта количества циклов длиной g, $g+2$ и $g+4$ с тем же порядком сложности, что умножение $n \times n$-матриц (при фиксированном g). В [8] сообщалось об аналогичном результате на основе упомянутых явных формул с ограничением $g \leqslant 12$. Работа [9] содержит описание алгоритма, который позволяет вычислять количество циклов длиной g, $g+2$, $\ldots$, $2\,g-2$ со сложностью $O(g\,m^2)$, где m — число рёбер в графе.

С увеличением обхвата графы неизбежно становятся более разреженными, вследствие чего для подсчёта циклов могут оказаться целесообразны перечислительные методы. В данной работе рассмотрены два таких метода: поиск с возвратом на основе [10] и способ, описанный в [11]. Второй способ интересен также тем, что допускает существенное ускорение при ограничении длины цикла значением $2\,g - 2$.

Целью данной работы стало исследование двух следующих вопросов, применительно к случаю двудольных графов.

1. Зависимость вычислительной сложности явных формул от учитываемого значения обхвата и длины подсчитываемых циклов. В частности, при каком их соотношении наиболее трудоёмкой операцией остаётся умножение матриц.
2. Сравнение по эффективности явных формул, алгоритмов [7], [9], [11] и поиска с возвратом на графах различной плотности. Особый интерес представляет случай наиболее плотных графов с заданным обхватом, так как авторы цитированных работ уделяли основное внимание только разреженным графам LDPC-кодов.

Основные результаты работы изложены в трёх разделах. В разделе 3 обсуждается учёт двудольности и обхвата в явных формулах для подсчёта циклов. Представлены характеристики выведенных выражений: количество слагаемых, порядки вычислительной сложности. Подробно исследуются случаи $k > 2\,g$ и $k \leqslant 2\,g$. Для первого случая указан класс форм маршрутов, приводящих к четырёхкратным суммам. Вместе с тем показано, что во втором случае наиболее трудоёмкой операцией оказывается умножение матриц. В завершение раздела рассматриваются количество матричных умножений и число самих матриц, требуемые при проведении вычислений по явным формулам.

Раздел 4 содержит анализ альтернативных алгоритмов подсчёта коротких циклов в двудольных графах: «леденцов» [7], «передачи сообщений» [9], «цепей» [11] и бэктрэкинга на основе [10]. В алгоритме «леденцов», как и при вычислениях по явным формулам, граф задаётся матрицей смежности. Все остальные методы основаны на представлении графа в виде списков смежности или инцидентности. Для оригинальной реализации алгоритма «леденцов» [12] приводятся точное количество матриц и матричных умножений, участвующих в расчётах. Метод «цепей» воплощён в двух вариантах: без ограничения длины цикла и с ограничением $2\,g-2$. В заключительном подразделе приводится сравнительный анализ порядков сложности алгоритмов при подсчёте циклов длиной не более $2\,g - 2$ в двудольных графах различной плотности.

Раздел 5 отведён для обсуждения вычислительных экспериментов. Приводится описание вычислительной среды и семейств графов, использовавшихся для тестирования методов подсчёта коротких циклов. Представлены зависимости затрат процессорного времени от порядка графа для алгоритмов, рассмотренных в предыдущих разделах. Вычисления выполнялись для графов различной плотности с обхватами 4, 6 и 8. В завершение раздела формулируются рекомендации по применению методов в зависимости от обхвата и плотности графа.

2. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

В основном, в работе используется терминология [13] и рассматриваются двудольные неориентированные конечные графы без петель и кратных рёбер.

Маршрутом длиной k называется упорядоченный набор $(v_1; v_2; \ldots; v_{k+1})$ вершин графа, такой что вершины v_i и v_{i+1} смежны. Цепь — это маршрут, в котором все вершины различны, а цикл есть маршрут длиной не менее 3, в котором все вершины кроме последней различны, а последняя совпадает с первой. При подсчёте циклы, отличающиеся только выбором начальной вершины или направления обхода вершин, рассматриваются как один.

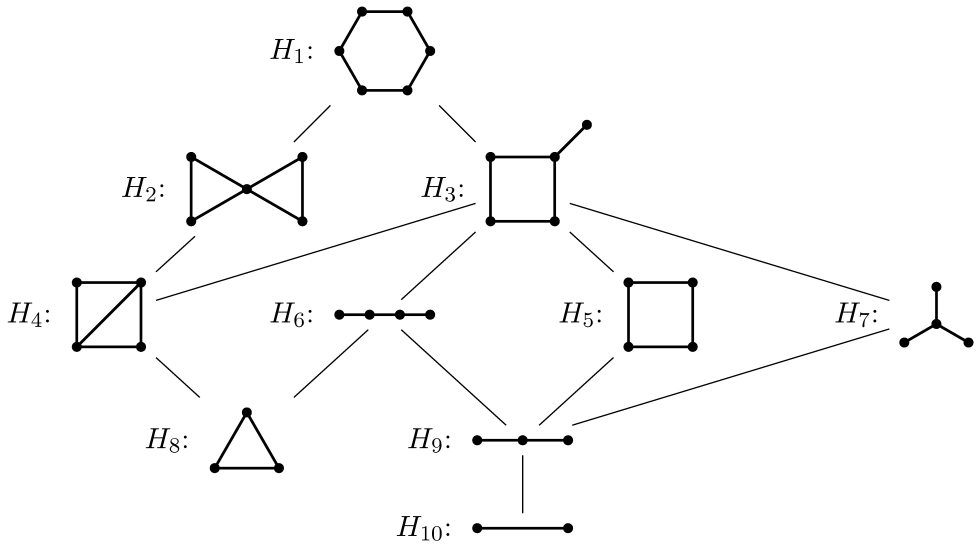

Рис. 1. Всевозможные формы замкнутых маршрутов длиной 6

Символы n, m, g и A закреплены за количеством вершин (порядком), количеством рёбер (размером), длиной кратчайшего цикла (обхватом) и матрицей смежности графа, в котором подсчитываются циклы. Доли двудольного графа обозначаются V_1 и V_2, а количество вершин в них — n_1 и n_2. Буквой k будем обозначать длину или наибольшую длину подсчитываемых циклов, а символом c_k — количество циклов длиной k. Граф, состоящий из одного простого цикла с k вершинами, обозначается C_k, а из одной простой цепи с k вершинами — P_k.

Элементы матрицы A записываются в виде a_{ij}, а элементы её степеней A^l — в виде $a_{ij}^{(l)}$. Диагональ матрицы A^2 состоит из степеней вершин графа, поэтому вместо $a_{ii}^{(2)}$ обычно применяется символ d_i. Матричные выражения, встречающиеся в работе, содержат следующие обозначения: «$\cdot$» или отсутствие знака — обычное умножение, «$\times$» — поэлементное умножение, «$^{\mathrm{T}}$» — транспонирование, $\mathrm{tr}(A)$ — след A.

Часто для множества целых чисел вида $\{a; a+1; \ldots; b\}$ используется обозначение $a\,..\,b$. Взятие целой и дробной частей числа a записывается с помощью скобок: $[a]$ и $\{a\}$.

3. ЯВНЫЕ ФОРМУЛЫ

3.1. Структура формул

Явные формулы для количества циклов длиной k, представленные в [3, 4, 5], основаны на перечислении всевозможных форм замкнутых маршрутов длиной k. Сопоставим с маршрутом подграф, состоящий из вершин и рёбер, по которым проходит маршрут. Например, циклу длиной k соответствует подграф, изоморфный C_k, а замкнутому маршруту длиной k, проходящему по единственному ребру, — подграф, изоморфный P_2. При этом один и тот же подграф может соответствовать нескольким маршрутам. Так, для каждого подграфа, изоморфного C_k, существуют $2\,k$ циклов длиной k, а для всякого подграфа, изоморфного P_2, — только два замкнутых маршрута чётной длины k. Отвлекаясь от обозначений вершин, будем говорить о графе, соответствующем маршруту, как о форме этого маршрута.

На рисунке 1 изображены всевозможные формы замкнутых маршрутов длиной 6. Каждая форма получается из C_6 путём серии попарных отождествлений вершин, в результате которых количество вершин уменьшается. Нумерация форм, указанная на рисунке 1, выполнена

в порядке невозрастания количества вершин. При одном и том же числе вершин форма меньшего размера имеет больший номер. Для других фиксированных значений k семейство форм упорядочивается аналогично. В частности, цикл C_k всегда имеет номер 1.

Пусть G — граф, в котором подсчитываются циклы. Зафиксируем длину цикла k и обозначим символом β_i количество подграфов графа G, изоморфных H_i. Для заданного номера i величина β_i, в частности $\beta_1 = c_k$, может быть выражена через матрицу смежности графа и величины β_j с определёнными номерами $j > i$. Например, в случае $k = 6$ получается следующая система [5, приложение C]:

$$\begin{gathered}
\beta_1 = \frac{1}{12}\sum_{i=1}^{n} a_{ii}^{(6)} - 2\,\beta_2 - \beta_3 - 3\,\beta_4 - 4\,\beta_5 - \frac{1}{2}\beta_6 - \beta_7 - 2\,\beta_8 - \beta_9 - \frac{1}{6}\beta_{10}, \\
\beta_2 = \frac{1}{8}\sum_{i=1}^{n}\left(a_{ii}^{(3)}\right)^2 - 2\,\beta_4 - \frac{3}{2}\beta_8, \\
\beta_3 = \frac{1}{2}\sum_{i=1}^{n} a_{ii}^{(4)} d_i - 2\,\beta_4 - 8\,\beta_5 - \beta_6 - 3\,\beta_7 - 3\,\beta_8 - 4\,\beta_9 - \beta_{10}, \\
\beta_4 = \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij}^{(2)}\right)^2 a_{ij} - \frac{3}{2}\beta_8, \quad \beta_5 = \frac{1}{8}\sum_{i=1}^{n} a_{ii}^{(4)} - \frac{1}{2}\beta_9 - \frac{1}{4}\beta_{10}, \\
\beta_6 = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{(3)} - 3\,\beta_8 - 2\,\beta_9 - \beta_{10}, \quad \beta_7 = \frac{1}{6}\sum_{i=1}^{n} d_i^3 - \beta_9 - \frac{1}{3}\beta_{10}, \\
\beta_8 = \frac{1}{6}\sum_{i=1}^{n} a_{ii}^{(3)}, \quad \beta_9 = \frac{1}{2}\sum_{i=1}^{n} d_i^2 - \beta_{10}, \quad \beta_{10} = \frac{1}{2}\sum_{i=1}^{n} d_i.
\end{gathered} \tag{1}$$

Система (1) и аналогичные ей системы для других значений k имеют треугольную структуру. Значение β_1, равное количеству циклов, находится путём последовательного вычисления β_i в порядке убывания номеров. Можно также явно выразить количество циклов через матрицу смежности, решив систему в символьном виде.

С ростом k количество форм резко увеличивается и при $k = 15$ уже равно 21190. Сами формулы к настоящему моменту выведены только для значений $k \leqslant 13$ [5]. Списки форм и Maple-код формул можно найти на сайте [14]. Применительно к различным семействам графов удаётся упростить выражения типа (1), исключив величины β_i, тождественно равные нулю на этих семействах. Способ вычисления коэффициентов при величинах β_i, предложенный в [5], позволяет учесть специфику графов в ходе самого вывода. За счёт этого в частных формулах можно продвинуться до бóльших значений длины цикла по сравнению с общим случаем.

3.2. Учёт двудольности

При подсчёте циклов в двудольных графах тождественно обращаются в нуль величины β_i для всех недвудольных форм H_i. Например, в случае $k = 6$ можно не рассматривать формы H_2, H_4 и H_8 (рисунок 1). При этом существенно сокращается как количество уравнений в системах вида (1), так и количество слагаемых в самих уравнениях, что позволяет вывести выражение для подсчёта циклов длиной 14. В то же время имеющиеся общие формулы дают выражения только для $k \leqslant 12$.

Двудольность можно учесть не только на этапе вычисления коэффициентов, но и при генерации самих форм. Для этого следует отождествлять только такие вершины C_k, расстояние между которыми чётно. За счёт сужения перебора удаётся продвинуться на несколько значений k дальше по сравнению с общим случаем. При этом, однако, принципиального ускорения

способа генерации, описанного в [5], не происходит, так как сохраняется скорость, с которой растёт количество проверок изоморфизма. В таблице 1 приводятся результаты вычислений с использованием системы компьютерной алгебры Maple 13, выполненные в рамках такого подхода. Пустые клетки соответствуют времени вывода менее 1 секунды. Прочерками отмечены

Таблица 1. Количество двудольных форм замкнутых маршрутов длиной k и время вывода явных формул для подсчёта циклов

k	4	6	8	10	12	14	16	18
Количество форм	3	7	20	59	230	1 002	5 308	31 709
Генерация форм (с)					2,3	32	880	45 940
Вычисление коэффициентов (с)				3,8	173	11 574	—	—

случаи, для которых расчёты не выполнялись.

Явные формулы для подсчёта циклов в большинстве случаев являются громоздкими и ориентированы преимущественно на компьютерную реализацию. Их несомненное достоинство состоит в достаточно простой структуре, которая позволяет эффективно распараллелить вычисления, обеспечивая линейное по числу процессоров ускорение [15].

Выражения для c_k с учётом двудольности не только компактнее общих формул, но и имеют на порядок меньшую сложность (кратность суммы) в случаях $k = 8, 10, 14$ [3, 4].

3.3. Учёт обхвата

В случае графов с обхватом не менее g можно исключить из явных формул для подсчёта циклов все величины β_i, для которых обхват H_i менее g. Коэффициенты в формулах позволяет вычислить общая процедура, описанная в [5], на вход которой следует подать список форм, отобранных с учётом обхвата. В таблицах 2 и 3 приводятся результаты вычислений для двудольных графов. Списки самих форм доступны на сайте [14]. Пустые клетки в таблице 3

Таблица 2. Количество двудольных форм замкнутых маршрутов длиной k с обхватом не менее g

$g \setminus k$	4	6	8	10	12	14	16	18
4	3	7	20	59	230	1 002	5 308	31 709
6		5	10	23	60	178	622	2 473
8			8	16	35	84	238	741
10				14	27	59	138	383
12					25	50	107	252
14						48	97	214
16							95	203
18								201

Таблица 3. Порядки сложности формул для подсчёта циклов длиной k в двудольных графах с обхватом не менее g

$g \setminus k$	10	12	14	16	18
4	4	6	6	8	8
6			4	4	6
8					4

соответствуют случаям, когда наиболее трудоёмкой операцией во всей формуле оказывается

умножение матриц. Исходя из представленных данных, можно предположить, что все такие случаи, и только они, определяются неравенством $k \leqslant 2\,g$. Следующие два подраздела посвящены обоснованию данного утверждения.

3.4. Случай $k > 2\,g$

Когда $k > 2\,g$, среди форм замкнутых маршрутов длиной k встречаются такие, которые приводят к суммам кратности не менее четырёх. Укажем способ построения одного класса форм, соответствующих четырёхкратным суммам. Первый представитель этого класса возникает при $k = 10$ и $g = 4$.

Выполним четыре отождествления вершин C_{10}, как указано на рисунке 2. Сначала из цикла

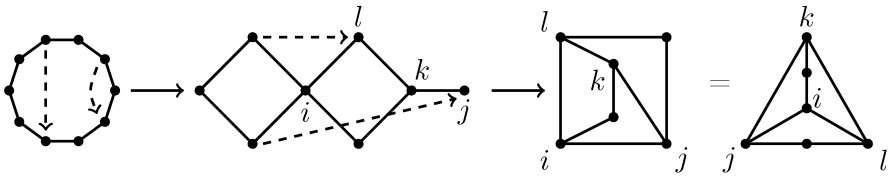

Рис. 2. Построение формы замкнутых маршрутов длиной 10, приводящей к четырёхкратной сумме

длиной 10 получаются два меньших цикла длиной 4 с одной общей вершиной и цепь длиной 1, присоединённая к одному из циклов на расстоянии 2 от общей вершины. Сомкнув два смежных ребра из разных циклов, получаем пару циклов с общим ребром $\{i; l\}$. Наконец, висячая вершина j отождествляется с наиболее удалённой от неё вершиной. Построенной форме соответствует следующая сумма [5]:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ij}a_{ik}^{(2)}a_{il}a_{jk}a_{jl}^{(2)}a_{kl}. \tag{2}$$

Два индекса, соответствующие непомеченным вершинам формы, могут быть исключены путём введения квадрата матрицы смежности. В полученной сумме каждый индекс встречается в парах с тремя другими, поэтому далее кратность суммы понизить не удаётся. Аналогичная ситуация возникает с формой K_4 при выводе выражения для количества циклов длиной 8 в произвольных графах.

Указанный способ построения формы, приводящей к четырёхкратной сумме, обобщается на бóльшие значения $k \equiv 2 \pmod 4$. Конечная форма имеет обхват $k/2 - 1$ и состоит из двух циклов длиной $k/2 - 1$ с общим ребром, к которым добавлено ребро, соединяющее наиболее удалённые вершины. Соответствующая сумма аналогична (2):

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ij}^{(p)}a_{ik}^{(q)}a_{il}a_{jk}a_{jl}^{(q)}a_{kl}^{(p)}, \quad p = (k-6)/4, \quad q = (k-2)/4.$$

Случай $k \equiv 0 \pmod 4$, $k \geqslant 12$, сводится к рассмотренному. В качестве исходной формы вместо C_k следует взять цикл C_{k-2}, который также является одной из форм замкнутых маршрутов длиной k. Тогда конечная форма будет иметь обхват $k/2 - 2$, который в случае $k \equiv 0 \pmod 4$ является наибольшим при условии $k > 2\,g$ (с учётом двудольности).

3.5. Случай $k \leqslant 2\,g$

Если $k \leqslant 2\,g$, все формы приводят к выражениям, в которых наиболее трудоёмкой операцией является обычное умножение матриц.

Независимо от обхвата, в состав форм входят все деревья порядков $2\,..\,k/2+1$. Кратности соответствующих им сумм всегда можно уменьшить вплоть до единичной за счёт использования вспомогательных векторов, так как всегда находится индекс, встречающийся в паре только с одним другим индексом [5]. Следующая последовательность преобразований иллюстрирует способ введения таких векторов на примере цепи P_4:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ij}a_{jk}a_{kl} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}d_k = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}v_j = \sum_{i=1}^{n} u_i,$$

где $v_j = \sum a_{jk}d_k$, а $u_i = \sum a_{ij}v_j$. В результате при фиксированном k время вычисления сумм для деревьев есть величина $O(n^2)$. Количество памяти, требуемой при этом для хранения вспомогательных векторов, составляет $O(n)$.

При $k = g$ кроме деревьев имеется единственная форма C_g. Если $k = g+2$, добавляются ещё две формы: C_{g+2} и форма, образованная присоединением к одной из вершин C_g новой вершины. Когда $k = g+4$, появляются также C_{g+4}, C_{g+2} «плюс вершина» и $g/2+2$ формы, получаемые присоединением к C_g двух новых вершин (вторая может присоединяться к первой). Кратности соответствующих сумм можно понизить до двух или одного, используя вспомогательные векторы и матрицы. Например, для формы, получаемой из C_4 присоединением к двум несмежным вершинам по одной новой вершине, имеем сумму

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{i'=1}^{n}\sum_{k'=1}^{n} a_{ij}a_{jk}a_{kl}a_{li}a_{ii'}a_{kk'} =$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{n}\left(\sum_{j=1}^{n} a_{ij}a_{jk}\right)\left(\sum_{l=1}^{n} a_{kl}a_{li}\right)d_i d_k = \sum_{i=1}^{n}\sum_{k=1}^{n} a_{ik}^{(2)}a_{ki}^{(2)}d_i d_k. \quad (3)$$

Аналогично упрощаются суммы для других форм, перечисленных выше:

$$\sum_{i=1}^{n} a_{ii}^{(l)},\ \sum_{i=1}^{n} a_{ii}^{(l)}d_i,\ \sum_{i=1}^{n} a_{ii}^{(k-4)}d_i^2,\ \sum_{i=1}^{n} a_{ii}^{(k-4)}\sum_{j=1}^{n} a_{ij}^{(2)},\ \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{(h)}a_{ij}^{(k-4-h)}d_i d_j. \quad (4)$$

В зависимости от соотношения k и g параметр l в первой сумме может принимать значения k, $k-2$ и $k-4$, а во второй — значения $k-2$ и $k-4$. Остальные выражения возникают в случае $k = g+4$. Параметр h в последней сумме пробегает значения с 1 по $k/2-2$. После указанных преобразований во всей формуле для подсчёта циклов длиной $k \leqslant g+4$ наиболее трудоёмкой операцией оказывается умножение матриц. Объём памяти, используемой в вычислениях, определяется количеством вспомогательных матриц и составляет $O(n^2)$.

При $k = 8$ и $g = 4$ есть четыре формы, которые содержат более одного цикла (рисунок 3). Им соответствуют суммы (5).

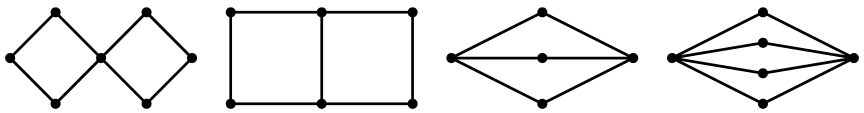

Рис. 3. Формы замкнутых маршрутов длиной 8, содержащие более одного цикла

$$\sum_{i=1}^{n}\left(a_{ii}^{(4)}\right)^2,\quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij}^{(3)}\right)^2 a_{ij},\quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij}^{(2)}\right)^3,\quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij}^{(2)}\right)^4. \quad (5)$$

Рассмотрим всевозможные формы замкнутых маршрутов длиной k с обхватом не менее $k/2$. Приводимое ниже рассуждение следует способу генерации форм [5], который основан на последовательном отождествлении вершин исходного цикла C_k. Формы, содержащие циклы нечётной длины, не принимаются во внимание, так как все формы, порождаемые ими, также содержат циклы нечётной длины. Если $k \equiv 2 \pmod 4$, то любая форма содержит не более одного цикла длиной не менее $k/2$ и приводит к сумме, кратность которой уменьшается вплоть до единичной за счёт использования вспомогательных матриц, так как всегда находится индекс, встречающийся в парах не более чем с двумя другими [5]. Помимо степеней матрицы смежности, которыми можно ограничиться в случае $k \leqslant g+4$, для упрощения некоторых сумм требуются и другие вспомогательные матрицы. Например, замкнутые маршруты длиной 16 могут иметь форму, которая получается из цикла C_8 присоединением к его вершинам, через одну, четырёх новых вершин. Этой форме соответствует сумма, подобная (3):

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a^{(2)}_{ij}a^{(2)}_{jk}a^{(2)}_{kl}a^{(2)}_{li}d_id_jd_kd_l = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} p_{ij}a^{(2)}_{jk}p_{kl}a^{(2)}_{li}d_id_k = \sum_{i=1}^{n}\sum_{k=1}^{n} q_{ik}q_{ki}d_id_k,$$

где $p_{ij} = a^{(2)}_{ij}d_j$, а $q_{ik} = \sum_j p_{ij}a^{(2)}_{jk}$.

При $k \equiv 0 \pmod 4$, $k \geqslant 8$, существует класс форм, содержащих более одного цикла длиной $k/2$ (рисунок 3). Первая форма получается при отождествлении двух вершин C_k, находящихся на расстоянии $k/2$ друг от друга, и представляет собой пару циклов с общей вершиной. Далее, смыкая смежные рёбра из разных циклов, получаем пары циклов с общими цепями длиной не более $k/4$. Ещё одна форма получается, если в паре циклов с одной общей вершиной отождествить вершины, находящиеся от общей на расстоянии $k/4$. Перечисленным формам с двумя и более циклами соответствуют следующие суммы:

$$\sum_{i=1}^{n}\left(a^{(k/2)}_{ii}\right)^2, \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a^{(k/2-h)}_{ij}\right)^2 a^{(h)}_{ij}, \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a^{(k/4)}_{ij}\right)^3, \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a^{(k/4)}_{ij}\right)^4.$$

Во второй сумме параметр h может принимать значения с 1 по $k/4-1$.

3.6. Реализация формул

Для каждой пары k и g, $k \geqslant g$, существует отдельный набор формул вида (1). К настоящему моменту выведены только наборы при $k \leqslant 18$, за исключением случаев $k = 16$, $g = 4$ и $k = 18$, $g = 4, 6$. Maple-код формул размещён на сайте [14]. При фиксированном значении обхвата набор выражений, соответствующий длине k, содержит все предыдущие наборы, и тем самым позволяет подсчитывать статистику циклов длиной не более k.

Для случая $k \leqslant \min\{2\,g; 18\}$ создана также C++-программа. Использование бинарной реализации позволяет более объективно сравнить эффективность явных формул с эффективностью другого матричного метода подсчёта циклов — алгоритма «леденцов» [7] ($k \leqslant g+4$), — для которого имеется оригинальная C++-реализация [12].

В C++-программе наборы формул вида (1) закодированы в «сокращённом» варианте. Каждая система разрешена относительно величин β_i и отобраны только выражения для количества циклов. Вспомогательные векторы и матрицы, входящие в несколько сумм, при вычислениях сохраняются в специальных массивах во избежание повторного расчёта.

Поскольку для явных формул и для алгоритма [7] наиболее трудоёмкой операцией является умножение матриц, основной интерес при их сравнении представляет количество выполняемых матричных умножений, а также объём памяти, требуемой для хранения промежуточных

Таблица 4. Количество матричных умножений при подсчёте циклов длиной не более k в двудольных графах с обхватом g

$g \setminus k$	g	$g+2$	$g+4$	$g+6$	$g+8$
4	2	3	5		
6	3	5	6	15	
8	6	7	9	22	91
10	6	8	12	31	144
12	9	10	15	42	
14	9	11	18		
16	12	13			
18	12				

Таблица 5. Количество одновременно хранящихся матриц при подсчёте циклов длиной не более k в двудольных графах с обхватом g

$g \setminus k$	g	$g+2$	$g+4$	$g+6$	$g+8$
4	9	12	18		
6	12	14	17	24	
8	13	14	20	28	50
10	12	14	23	29	57
12	13	14	26	34	
14	12	14	29		
16	13	14			
18	12				

результатов (матриц). Указанные в таблицах 4 и 5 данные получены с помощью дополнительного кода, помещённого в программу для подсчёта циклов.

Элементы всех матриц, участвующих в явных формулах, представляют количество маршрутов определённого вида. Например, элемент $a_{ij}^{(l)}$ есть количество всех маршрутов длиной l, соединяющих вершины с номерами i и j [13]. Учитывая двудольность графов, можно оптимизировать выполнение матричных операций, представив матрицу смежности и матрицы маршрутов в блочном виде. Такой вид соответствует нумерации вершин доли V_1 числами с 1 по n_1, а вершин доли V_2 — числами с n_1+1 по n. Нулевые и ненулевые блоки меняются местами при смене чётности длины маршрута. В частности, последовательность степеней матрицы смежности имеет вид:

$$A = \begin{pmatrix} 0 & B \\ B^{\mathrm{T}} & 0 \end{pmatrix}, \; A^2 = \begin{pmatrix} BB^{\mathrm{T}} & 0 \\ 0 & B^{\mathrm{T}}B \end{pmatrix}, \; A^3 = \begin{pmatrix} 0 & BB^{\mathrm{T}}B \\ B^{\mathrm{T}}BB^{\mathrm{T}} & 0 \end{pmatrix}, \; \ldots. \tag{6}$$

Подматрицы степеней A с нечётными показателями взаимно транспонированны, поэтому достаточно хранить и вычислять по одной из них. Данные в таблицах 4 и 5 относятся именно к блокам матриц, без учёта их размеров.

Рассмотрим зависимость количества умножений и вспомогательных матриц от обхвата графа g при подсчёте циклов длиной g, $g+2$ и $g+4$. В следующем разделе аналогичная информация приводится для алгоритма [7].

Умножение матриц необходимо возникает в суммах, которые соответствуют формам, содержащим циклы. В результате получаются выражения вида (4) и (5). При этом в качестве промежуточных результатов требуется вычислить все подряд степени матрицы смежности с A^2 по A^{g+4}. За счёт увеличения кратности суммирования можно понизить наибольший пока-

затель степени. Например, для первой суммы (4) имеем:

$$\sum_{i=1}^{n} a_{ii}^{(l)} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{(l/2)} a_{ji}^{(l/2)} = \sum_{i=1}^{n}\sum_{j=1}^{n} \left(a_{ij}^{(l/2)}\right)^2. \tag{7}$$

Аналогичное преобразование следующих трёх сумм (4) приводит к выражениям

$$\sum_{i=1}^{n} d_i \sum_{j=1}^{n} \left(a_{ij}^{(l/2)}\right)^2, \quad \sum_{i=1}^{n} d_i^2 \sum_{j=1}^{n} \left(a_{ij}^{(g/2)}\right)^2, \quad \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \left(a_{ij}^{(g/2)}\right)^2\right) \left(\sum_{j=1}^{n} \left(a_{ij}^{(2)}\right)\right).$$

В результате, если $g > 4$, исчезают степени A^g, …, A^{g+4}, участвовавшие только в первых четырёх суммах (4) до преобразования. При $g = 4$ уже упрощённая сумма (7), когда $l = 8$, содержит матрицу A^4. Избавляться подобным образом от степеней с показателями $g - 1$ и менее, которые входят в последнюю сумму (4), нецелесообразно, так как будут возникать трёхкратные суммы, вычисляемые не быстрее умножения матриц. В случае $g = 4$ аналогичное замечание относится к дальнейшему «упрощению» суммы (7) при $l = 8$.

С учётом предложенного преобразования сумм в формулах для подсчёта циклов длиной g, $g+2$ и $g+4$ остаются только степени A^2, …, A^{g-1} (при $g = 4$ ещё A^4). Все умножения матриц выполняются при вычислении этих степеней.

Следуя схеме (6), нетрудно подсчитать количество блоков разного размера и количество их умножений. Матрицы A^2 и A^3 получаются из матрицы смежности путём трёх умножений: BB^{T}, $B^{\mathrm{T}}B$ и $(BB^{\mathrm{T}})B$. Аналогично, с теми же вычислительными затратами, путём домножения на блоки B и B^{T} можно получить остальные пары матриц, заканчивая A^{g-2} и A^{g-1}.

Таким образом насчитывается по $(g-2)/2$ умножений каждого из трёх типов: $(n_1 \times n_2)\cdot(n_2 \times n_1)$, $(n_2 \times n_1)\cdot(n_1 \times n_2)$ и $(n_1 \times n_1)\cdot(n_1 \times n_2)$. Соответственно накапливаются по $(g-2)/2$ матриц с размерами $n_1 \times n_1$, $n_2 \times n_2$ и $n_1 \times n_2$, в дополнение к исходному блоку B. В случае $g = 4$, для вычисления A^4, необходимо дополнительно сделать по одному умножению $(n_1 \times n_2) \cdot (n_2 \times n_1)$ и $(n_2 \times n_1) \cdot (n_1 \times n_2)$ и сохранить две матрицы с размерами $n_1 \times n_1$ и $n_2 \times n_2$.

Практически помимо матриц степеней используются ещё 10 матриц (12 при $g = 4$) (таблица 5), в число которых входят поэлементные произведения. Например, сумма $\sum_{i,j} \left(a_{ij}^{(2)}\right)^2$ представляется в виде $\sum_{i,j}(A^2 \times A^2)_{ij}$. К «лишним» матрицам относятся также локальные переменные и временные объекты, связанные с особенностями вычисления выражений в программе на языке C++. Общее количество элементов блоков и умножений блоков приводится в таблице 6. Символом t_{pqr}, $p, q, r \in \{1; 2\}$, обозначается временна́я сложность умножения матриц с размерами $n_p \times n_q$ и $n_q \times n_r$. Случаи $g = 6, 8$ выписаны отдельно для удобства сравнения с таблицей 7.

Таблица 6. Количество матричных умножений и элементов одновременно хранящихся блоков при подсчёте циклов длиной не более $g + 4$

g	Количество умножений	Количество элементов
4	$t_{112} + 2(t_{121} + t_{212})$	$7(n_1^2 + n_2^2) + 4\,n_1 n_2$
6	$2(t_{112} + t_{121} + t_{212})$	$2(n_1^2 + n_2^2) + 13\,n_1 n_2$
8	$3(t_{112} + t_{121} + t_{212})$	$4(n_1^2 + n_2^2) + 12\,n_1 n_2$
> 4	$(g/2 - 1)(t_{112} + t_{121} + t_{212})$	$2[g/4](n_1^2 + n_2^2) + (g/2 + 4\{g/4\} + 8)n_1 n_2$

При использовании универсального правила умножения матриц затраты памяти и времени, требуемые для расчётов по явным формулам, не зависят от плотности графа. Однако ряд

формул обладает той особенностью, что во всех выполняемых умножениях участвует матрица смежности. Например, таковы все формулы, рассмотренные выше при условии $k \leqslant g + 4$.

Умножение произвольной матрицы A на матрицу B, состоящую из нулей и единиц, можно выполнить более эффективно, вычисляя элемент произведения c_{ij} не как сумму $\sum a_{ik} b_{kj}$, а как сумму $\sum_{l \in L_j} a_{ik_l}$, где L_j — множество всех значений k, для которых $b_{kj} = 1$. Вычислительная сложность такого способа умножения является величиной $O(m \sum |L_j|)$, где m — количество строк матрицы A.

Построение множеств L_j требует дополнительных вычислительных ресурсов, однако эти затраты меньше, чем сложность «обычного» умножения матриц. Более того, в случае с формулами для количества циклов множества L_j находятся один-единственный раз — для матрицы смежности. При сравнении различных алгоритмов в следующих разделах рассматриваются оба варианта формул: с использованием общего правила умножения матриц и с учётом разреженности.

Если граф достаточно разрежен, имеет смысл представить умножение на степень матрицы смежности в виде последовательности умножений на матрицу смежности. Так, в формулу для подсчёта циклов длиной 14 в двудольных графах с обхватом 8 входят два слагаемых, при вычислении которых выполняется умножение на квадрат матрицы смежности. Указанный приём позволил понизить порядок вычислительной сложности этой формулы для семейств разреженных графов (как показано далее в разделе 5).

4. ДРУГИЕ АЛГОРИТМЫ

4.1. Алгоритм «леденцов»

Алгоритм [7] предназначен для подсчёта циклов длиной g, $g+2$ и $g+4$ в двудольном графе с обхватом g, который также определяется в ходе вычислений. Основу метода составляют матричные операции, однако в отличие от явных формул, представленных в предыдущем разделе, пересчитывается фиксированный набор матриц, не зависящий от значения обхвата. Ещё одно отличие состоит в том, что алгоритм [7] изначально оперирует матрицами, в которых строки соответствуют вершинам одной доли графа, а столбцы — вершинам другой доли.

Пусть B — ненулевой блок матрицы смежности, согласно (6). Введём матрицы P_k^1 количества цепей длиной k с началом в V_1. Если k нечётно, цепи оканчиваются в V_2 и матрица P_k^1 имеет размер $n_1 \times n_2$, а если чётно, то концы цепей принадлежат V_1 и размер матрицы есть $n_1 \times n_1$. Аналогично определяются матрицы P_k^2. Для количества циклов длиной k имеет место выражение

$$c_k = \frac{1}{k} \operatorname{tr}(P_{k-1}^1 B^{\mathrm{T}}) = \frac{1}{k} \operatorname{tr}(P_{k-1}^2 B).$$

Деление на k выполняется в связи с тем, что в подсчёте участвуют циклы, отличающиеся выбором начальной вершины из определённой доли ($k/2$ вариантов) и направления обхода вершин (2 варианта). Матрица P_k^1 рассчитывается с помощью P_{k-1}^1 и матриц $L_{k-2\,l;2\,l}^1$ количества $(k-2\,l;2\,l)$-«леденцов», $l = 2, 4, \ldots, 2[k/2]$. «Леденцом» (lollipop) называется маршрут длиной k, в котором все вершины кроме последней различны, а последняя совпадает с $(k-2\,l+1)$-й. В случае $l > 1$ «леденец» состоит из цепи длиной $k-2\,l$ и цикла длиной $2\,l$, имеющих одну общую вершину (рисунок 4). Если же $l = 1$, то «леденец» проходит дважды по одному ребру. Матрицы «леденцов» определяются аналогично матрицам цепей.

При нечётном k справедливо соотношение:

$$P_k^1 = P_{k-1}^1 B - L_{1;k-1}^1 - L_{3;k-3}^1 - \ldots - L_{k-2;2}^1. \tag{8}$$

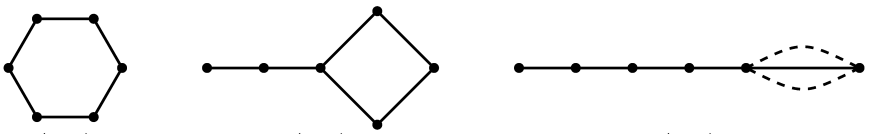

Рис. 4. Примеры «леденцов» длиной 6

Сами матрицы «леденцов» можно выразить, используя тот же принцип. Так,

$$L^1_{g-1;2} = BL^2_{g-2;2} - P^1_{g-1} \times B - \max\{L^1_{0;2} - 1; 0\}L^1_{g-3;2} \tag{9}$$

(взятие максимума и вычитание единицы в последнем слагаемом выполняются поэлементно). Однако в случае выражений для матриц «леденцов» вычитаемые величины соответствуют разнообразным маршрутам, уже не являющимся «леденцами». Для них авторы [7] вывели индивидуальные явные формулы, в том числе (9), используя блок матрицы смежности, матрицы цепей и «леденцов» и различные матричные операции, выполняемые поэлементно, за исключением обычного умножения. В общем случае с ростом длины подсчитываемых циклов количество требуемых формул также неограниченно увеличивается согласно уравнений вида (8). Однако для подсчёта циклов длиной g, $g+2$ и $g+4$ достаточно постоянного конечного набора формул, так как многие матрицы «леденцов», входящие в формулы наподобие (8), тождественно обращаются в нуль. В [7] приведены выражения для матриц

$$L^1_{1;2},\ L^1_{2;2},\ L^1_{k;2}\ (k \leqslant g-2),\ L^1_{g-1;2},\ L^1_{g;2},\ L^1_{g+1;2},\ L^1_{1;g},\ L^1_{2;g},\ L^1_{3;g},\ L^1_{1;g+2},$$

вывод которых содержится в отдельной работе [12]. Расширение имеющегося набора формул для бóльших значений длины затрудняется громоздкостью новых выражений.

В оригинальной реализации [12] алгоритма «леденцов» совокупность рекуррентных соотношений из [7] представлена в виде итеративной вычислительной схемы. Последовательно находится количество циклов длиной k, начиная с 4, пока значение k не достигнет обхвата g или не превысит $k_{\max} = 2\min\{n_1; n_2\}$ (в коде [12] ошибочно запрограммирован max). Случаи $g = 4, 6, 8$, составляющие «базу» рекурсии, обрабатываются отдельно, а начиная с $g = 10$ вычисления принимают универсальный вид, при котором пересчитывается фиксированный набор матриц. Заключительная часть работы алгоритма состоит в подсчёте циклов длиной $g + 2$ и $g + 4$. Возможны, хотя и не типичны, особые случаи, когда $g + 4 > k_{\max}$. В таких ситуациях часть расчётов оказывается излишней и не выполняется.

Указанная итеративная реализация метода характеризуется линейным относительно g ростом количества матричных умножений и фиксированным объёмом требуемой памяти. В [7] приводятся следующие оценки: $6\,g + 37$ умножений при $g > 6$ и не более $11\,m^2 + 11\,n^2 + 21\,m\,n$ элементов одновременно хранящихся блоков. По причине громоздкости выражений, входящих в алгоритм «леденцов», вручную оценивать его сложность затруднительно. С помощью вставки в основные вычислительные процедуры дополнительного кода были получены детальные экспериментальные оценки, отличающиеся от оригинальных как в лучшую, так и в худшую стороны. В таблице 7 используются те же обозначения, что в таблице 6. Уникальность случаев $g = 4, 6, 8$ обусловлена их отдельной обработкой в программе.

В таблице 8 приводятся характеристики явных формул и алгоритма «леденцов», без учёта размеров матриц для наглядности. Затраты памяти для явных формул достигают того же уровня, что для алгоритма «леденцов», только при обхвате 20, и затем становятся больше.

Алгоритм «леденцов», как и некоторые явные формулы, допускает учёт разреженности графа (раздел 3.6).

Таблица 7. Количество матричных умножений и элементов одновременно хранящихся блоков при подсчёте циклов длиной не более $g+4$

g	Количество умножений	Количество элементов
4	$t_{111}+22\,t_{112}+19\,t_{122}+t_{222}$	$9\,n_1^2+19\,n_1n_2+10\,n_2^2$
6	$2\,t_{111}+22\,t_{112}+18\,t_{122}+2\,t_{222}$	$7\,n_1^2+25\,n_1n_2+8\,n_2^2$
8	$3\,t_{111}+33\,t_{112}+29\,t_{122}+3\,t_{222}$	$8\,n_1^2+29\,n_1n_2+9\,n_2^2$
>8	$(g/2-1)(t_{111}+t_{222})+(5\,g/2+4)(t_{112}+t_{122})+4\,t_{112}$	$8\,n_1^2+21\,n_1n_2+9\,n_2^2$

Таблица 8. Количество матричных умножений (слева) и количество блоков матриц (справа) при подсчёте циклов длиной не более $g+4$ в двудольных графах с обхватом g

g	Формулы	«Леденцы»
4	5	43
6	6	44
8	9	68
>8	$1{,}5\,g-3$	$6\,g+10$

g	Формулы	«Леденцы»
4	18	38
6	17	40
8	20	46
>8	$1{,}5\,g+8$	38

4.2. Алгоритм «передачи сообщений»

Алгоритм [9] (message-passing algorithm) позволяет находить количество циклов длиной g, $g+1,\ldots,2\,g-1$ в графе с известным обхватом g. В случае двудольных графов можно подсчитать циклы с длинами g, $g+2,\ldots,2\,g-2$. В этом методе не используются матрицы маршрутов, как в алгоритме «леденцов» или в явных формулах. Сложность алгоритма «передачи сообщений» зависит от плотности графа: по оригинальной оценке, время работы является величиной $O(g\,m^2)$, а количество памяти — величиной $O(m\cdot\min\{d_1;d_2\})$, где символами d_1 и d_2 обозначены наибольшие степени вершин в долях графа.

Главная идея метода состоит в подсчёте только тех маршрутов, для которых любые два «соседних» ребра различны. Замкнутые маршруты такого вида либо являются циклами, либо «содержат» более одного цикла (в частности, исключены «леденцы», отличные от циклов). Если длина маршрута меньше удвоенного обхвата, то единственным возможным вариантом остаётся цикл. Для указанного вида маршрутов нетрудно организовать переход от одного значения длины к следующему. Зафиксируем ребро $e=\{v_1;v_2\}$ и обозначим $w_{uv}^{(k)}$ (u и v — смежные вершины) количество маршрутов вида $(v_1;v_2;\ldots;v_{k+1})$, где $v_k=u$, $v_{k+1}=v$ и $v_i\neq v_{i+2}$ для всех $i\in 1\,..\,k-1$. Тогда

$$w_{uv}^{(k+1)}=\sum_{v'\in N_u\setminus\{v\}} w_{v'u}^{(k)},\qquad w_{uv}^{(1)}=\begin{cases}1, & u=v_1,\ v=v_2,\\ 0, & \text{иначе},\end{cases}\tag{10}$$

где N_u — множество всех вершин, смежных с u. При $k\leqslant 2\,g-2$ значение $w_{v_1v_2}^{(k+1)}$ в точности равно количеству циклов длиной k, проходящих через ребро e (независимо от того, какая из конечных вершин этого ребра выбрана в качестве v_1, а какая — в качестве v_2). Располагая величинами $w_{v_1v_2}^{(k+1)}$ для всех рёбер $\{v_1;v_2\}$, можно найти количество всех циклов длиной k:

$$c_k=\frac{1}{k}\sum_{\{v_1;v_2\}\in E} w_{v_1v_2}^{(k+1)},\quad k=4,6,\ldots,2\,g-2,$$

где E — множество всех рёбер. Значение суммы делится на k, так как каждый цикл длиной k встречается ровно k раз (по числу рёбер в нём).

В ходе вычислений по формуле (10) представляют интерес только ненулевые величины $w_{uv}^{(k)}$. На первом шаге ($k=1$) такое значение единственно, на втором шаге их количество

равно $|N_{v_2}| - 1$. В худшем случае с ростом k число ненулевых величин $w_{uv}^{(k)}$ достигает размера графа. При этом достаточно хранить только данные для текущего значения k. Следовательно объём требуемой памяти оценивается как $O(m)$, в дополнение к размеру исходных данных. Сам граф удобно хранить в виде списков инцидентности.

В оригинальной версии алгоритма вычисления по формуле (10) выполняются не для одного начального ребра $e = \{v_1; v_2\}$, а одновременно для всех рёбер, инцидентных выбранной вершине v_1. Соответственно величины $w_{uv}^{(k)}$ представляют собой векторы с $|N_{v_1}|$ компонентами, по одной для каждого ребра. В результате оценка объёма памяти достигает $O(m \cdot \min\{d_1; d_2\})$. При этом время работы теоретически сохраняется: меняются местами цикл по значениям k и цикл по начальным рёбрам.

Авторы [9] учли возможность оптимизации алгоритма: после того как подсчитаны циклы, проходящие через вершину $v_1 \in V_1$, следует пропускать эту вершину во всех операциях (другими словами, удалить её из графа). В таком случае каждый цикл длиной k рассматривается не k раз (по числу вершин в цикле из одной доли графа и числу направлений обхода), а только 2 раза (по числу направлений обхода).

При отдельной обработке начальных рёбер, как предложено выше, можно удалять начальное ребро e сразу после его обработки. Тогда все циклы учитываются по одному разу.

Время работы алгоритма существенно зависит от плотности графа. При достаточно высокой плотности количество пересчитываемых данных $w_{uv}^{(k)}$ быстро приближается к размеру графа для большой части начальных рёбер e (по мере обработки рёбра удаляются). Так, в полном двудольном графе время работы достигает величины порядка n^4 (худший случай), превышающей сложность умножения матриц.

Если же граф разрежен, то для каждого начального ребра вычисления по формуле (10) «распространяются» на относительно небольшое число пар $(u; v)$. Например, при фиксированной верхней границе для степеней вершин количество таких пар не зависит от порядка графа и время работы алгоритма составляет $O(g\,n)$. Данное значение не получается из общей оценки $O(g\,m^2)$ путём формальной подстановки $m = O(n)$.

Далее будем считать обхват фиксированным и опускать его в оценках сложности в качестве коэффициента. Учитывая случай графов с небольшими степенями вершин, целесообразно рассматривать уточнённую оценку времени работы алгоритма: $O(\min\{m^2; n_1 \cdot (d_1 d_2)^{g-1}\})$.

Алгоритм «передачи сообщений» не позволяет подсчитывать циклы длиной $2\,g$, тогда как в случае явных формул это значение ещё попадает в диапазон, для которого наиболее трудоёмкой операцией является умножение матриц.

4.3. Алгоритм «цепей»

В работе [11], посвящённой конструированию и исследованию LDPC-кодов, предложен метод поиска небольших «ловушек» в графе кода — подграфов, наличие которых приводит к ошибкам декодирования. Составной частью этого метода является процедура генерации всех циклов чётной длины k, проходящих через заданную вершину v, основанная на построении дерева цепей длиной $k/2 - 1$, начинающихся в v. Авторы [11] не приводят каких-либо аргументов в пользу именно такого способа генерации циклов по сравнению с другими методами (например, бэктрэкингом).

Сама процедура перечисления циклов описывается в [11] недостаточно подробно. В частности, не указана полная оценка её вычислительной сложности. Ниже мы приводим свою версию метода генерации (и, тем самым, подсчёта) циклов на основе дерева цепей, называемую впоследствии первым вариантом алгоритма «цепей». Рассматривая только задачу подсчёта, можно радикально уменьшить время работы алгоритма, если ограничить длины циклов зна-

чением $2\,g-2$, как в алгоритме «передачи сообщений». Соответствующая модификация будет называться вторым вариантом алгоритма «цепей».

Путём поиска в ширину из заданной вершины v_0 строится дерево цепей высоты $k/2-1$, так что вершины дерева, образующие уровень l (нумерация — с нуля), соответствуют всевозможным цепям длиной l с началом v_0 в исходном графе. Далее находятся все пары листьев построенного дерева, для каждой из которых соответствующие цепи образуют цикл при соединении их концов с какой-либо общей смежной вершиной v. Необходимо проверить, чтобы обе цепи не «пересекались», а также не проходили через вершину v. В [11] предписывается перебирать только пары листьев из разных поддеревьев с корнями, составляющими первый уровень дерева. В противном случае цепи заведомо «пересекаются».

Дерево цепей высоты $k/2-1$ позволяет найти не только циклы длиной k, но и более короткие циклы. Для поиска циклов длиной $l<k$ можно воспользоваться уровнем $l/2$ (вместо $l/2-1$) и рассматривать пары вершин этого уровня, соответствующих цепям с общим концом. С целью унификации процедуры поиска циклов различной длины дерево цепей удобно строить до высоты $k/2$.

Подсчёт всех циклов длиной не более k осуществляется путём запуска указанной процедуры для каждой вершины одной доли графа. Здесь также применим общий оптимизационный приём: не рассматривать вершину после того как найдены все циклы, проходящие через неё.

Оценим сложность метода при фиксированном k, используя величины d_1 и d_2, равные наибольшим степеням вершин из множеств V_1 и V_2 соответственно. Для определённости будем считать, что деревья цепей строятся для вершин доли V_1. Корень дерева соединён максимум с d_1 поддеревьями высотой $k/2-1$. На уровне l каждого поддерева насчитывается не более $(d_1-1)^{\lfloor l/2\rfloor}(d_2-1)^{\lceil l/2\rceil}$ вершин. Символами $\lfloor a\rfloor$ и $\lceil a\rceil$ обозначены наибольшее целое, не большее a, и наименьшее целое, не меньшее a. При фиксированном k данная оценка имеет наибольший порядок для последнего уровня и является величиной

$$O\left((d_1-1)^{\lfloor (k/2-1)/2\rfloor}(d_2-1)^{\lceil (k/2-1)/2\rceil}\right)=O\left(d_1^{\lceil k/4\rceil-1}d_2^{\lfloor k/4\rfloor}\right).$$

Время работы алгоритма и объём требуемой памяти, определяемые временем проверки пар листьев и размером всего дерева, соответственно, составляют

$$O\left(n_1 d_1^{2\lceil k/4\rceil}d_2^{2\lfloor k/4\rfloor}\right)\quad\text{и}\quad O\left(d_1^{\lceil k/4\rceil}d_2^{\lfloor k/4\rfloor}\right).$$

В частности, при $k=2\,g-2$ имеем оценки

$$O\left(n_1 d_1^{g}d_2^{g-2}\right)\quad\text{и}\quad O\left(d_1^{g/2}d_2^{g/2-1}\right).$$

Рассмотрим второй вариант метода, позволяющий существенно быстрее подсчитывать циклы длиной не более $2\,g-2$. При таком ограничении упрощается процедура построения дерева цепей: поскольку длины цепей не превышают $g-1$, то при наращивании цепи $(v_0;u;\ldots;w;v)$ путём добавления вершины, смежной с v, достаточно проверить, чтобы она не совпадала с w. Данная оптимизация не сказывается на порядке сложности процедуры построения дерева (и всего алгоритма). Главный же эффект состоит в том, что аналогичным образом упрощается проверка, образуют ли две цепи с общими концами v_0 и v цикл: достаточно сравнить только их вторые (u) и предпоследние (w) вершины.

Наличие только двух параметров u и w (при фиксированных v_0 и v) позволяет эффективно организовать подсчёт цепей, соответствующих различным парам $(u;w)$, избежав попарной проверки цепей. Обозначим p_{uw} количество цепей длиной $k\leqslant g-1$, имеющих вид $(v_0;u;\ldots;w;v)$.

Тогда число циклов длиной $2\,k$, разбиваемых в вершинах v_0 и v на пары цепей с длинами k, выражается по формуле

$$\sum_{\substack{u,w,u',w',\\ u\neq u',w\neq w'}} p_{uw}p_{u'w'} = \sum_{u,w} p_{uw}\cdot(p_{**} - p_{u*} - p_{*w} + p_{uw}), \tag{11}$$

где $p_{**} = \sum_{u,w} p_{uw}$, $p_{u*} = \sum_w p_{uw}$, а $p_{*w} = \sum_u p_{uw}$. За счёт избавления от перебора пар листьев время подсчёта (а не построения!) циклов для фиксированной начальной вершины v_0 сопоставимо с размером дерева. Время работы всего алгоритма при этом оказывается величиной

$$O\left(n_1 d_1^{g/2} d_2^{g/2-1}\right).$$

Алгоритмы «передачи сообщений» и «цепей» выполняют схожие построения, основанные на поиске в ширину. В первом методе на каждом уровне поиска аккумулируется количество маршрутов соответствующей длины, «оканчивающихся» различными рёбрами. Во втором же методе явно сохраняется каждая цепь, хотя глубина поиска вдвое меньше. Соответственно объём данных, обрабатываемых на каждом уровне в алгоритме «передачи сообщений», ограничен размером графа, тогда как количество цепей при достаточной плотности графа может быстро превысить и эту величину, и число матричных элементов, рассчитываемых в явных формулах. Если же граф разрежен настолько, что поиск распространяется только на «небольшую окрестность» начальной вершины, то за счёт меньшей глубины алгоритм «цепей» (особенно его версия без попарных сравнений) может оказаться более предпочтительным.

4.4. Поиск с возвратом

Существуют различные алгоритмы поиска с возвратом (backtracking) для перечисления всех циклов в графе. Некоторые из них, в которых длина цикла «жёстко» связана с глубиной поиска, такие как [10], нетрудно модифицировать для случая только коротких циклов.

При помощи поиска в глубину для каждой вершины выбранной доли графа строятся всевозможные цепи длиной $k-1$. На промежуточных шагах автоматически получаются все цепи меньшей длины. Отслеживая случаи, когда конец цепи смежен с началом, можно подсчитать циклы длиной не более k. Начальные вершины построенных цепей исключаются из графа с целью ускорения работы алгоритма. Дополнительная оптимизация достигается, если концы цепей длины $k-1$ проверять на смежность с началом не путём перебора смежных вершин, а обращаясь к соответствующей строке матрицы смежности. Хранить всю матрицу при этом не требуется: нужная строка создаётся из списка смежных вершин перед запуском поиска из начальной вершины.

Время выполнения алгоритма оценивается величиной

$$O\left(n_1 d_1^{\lceil (k-1)/2\rceil} d_2^{\lfloor (k-1)/2\rfloor}\right) \quad \left(O\left(n_1 d_1^{g-1} d_2^{g-2}\right) \text{ при } k = 2\,g-2\right)$$

по аналогии с алгоритмом «цепей». Объём же требуемой памяти есть $O(n)$, в дополнение к объёму входных данных.

При построении циклов длиной не более $2\,g-2$ наиболее простой способ учесть обхват — заботиться только о том, чтобы маршрут не проходил по какому-либо ребру более одного раза подряд, (как в алгоритме «передачи сообщений») и игнорировать другие «самопересечения» маршрута. В таком случае замкнутые маршруты «естественным» образом окажутся циклами. Данный приём упрощает структуру алгоритма, однако приводит к дополнительным ветвям поиска, которые заведомо не дают циклов.

Поиск с возвратом, как и первый вариант алгоритма «цепей», является перечислительным методом. Относительно времени его работы имеют место те же замечания. Радикальное отличие состоит в количестве требуемой памяти, которое в данном случае сопоставимо с объёмом входных данных. Выигрыш в памяти сопровождается дополнительными временны́ми затратами: «вторая половина» цикла всякий раз строится заново (поиск в два раза глубже).

4.5. Сравнение порядков сложности алгоритмов

Вычислительные затраты представленных выше алгоритмов выражены через разные параметры входного графа: порядок, размер, наибольшую степень вершины. Оценки, связанные с последней величиной, наиболее убедительны, когда разброс степеней невелик, поэтому сравнение методов удобно выполнить в случае регулярных графов. Остаются два независимых параметра — порядок и размер, — соотношение которых выразим через «плотность» графа ρ: $m = O(n^{1+\rho})$. Будем считать, что графы связны и $\rho \geqslant 0$. Введённые предположения позволяют представить сложность каждого метода в виде $O(n^{f(\rho;g;k)})$, где g — обхват графа, а k — длина цикла (при фиксированных ρ, g и k). Значения показателя f для рассмотренных алгоритмов приводятся в таблице 9.

Таблица 9. Порядки оценок времени работы и количества требуемой памяти

	фор.	фор. 2	«сообщения»	«цепи»	«цепи» 2	бэктрэкинг
Время	3	$2+\rho$	$\min\{2(1+\rho); \max\{2; 1+2\,(g-1)\rho\}\}$	$1+k\rho$	$1+(g-1)\rho$	$1+(k-1)\rho$
Память	2		$1+\rho$	$\max\{k\rho/2; 1+\rho\}$		$1+\rho$

При использовании явных формул и алгоритма «передачи сообщений» значение обхвата удобно задавать в качестве входного параметра. Процедура поиска кратчайшего цикла в двудольном графе, предложенная в [16], позволяет вычислить обхват за время $O(n^2)$. Объём требуемой памяти при этом составляет $O(n)$ (в дополнение к объёму входных данных $O(m)$). Наличие операции $\max\{2;\ldots\}$ в случае алгоритма «передачи сообщений» (таблица 9) связано именно с использованием указанной процедуры.

Сравнивая величины, приведённые в таблице 9, необходимо учитывать длину подсчитываемых циклов. За основу примем ограничение алгоритма «передачи сообщений»: $k \leqslant 2\,g-2$. Выведенные явные формулы позволяют подсчитывать циклы такой длины при $g \leqslant 10$, а алгоритм «леденцов» — при $g \leqslant 6$. В силу оценки $\rho \leqslant 2/(g-2)$ для двудольных графов (которая точна, по крайней мере, при $g = 4, 6, 8, 12$) [17] порядки временной сложности всех рассмотренных алгоритмов при выбранном ограничении длины цикла не превышают 7.

Соотношение оценок, представленных в таблице 9, зависит от «плотности» графа. Количество требуемой памяти может существенно превосходить объём входных данных в случае методов, использующих матрицы маршрутов (для разреженных графов) или дерево цепей (для плотных графов). Когда $\rho \geqslant 2/(k-2)$, размер дерева цепей может оказаться больше размера исходного графа. Если же плотность ещё выше, $\rho \geqslant 4/k$, то для хранения дерева цепей может потребоваться больше памяти, чем для хранения матриц маршрутов.

При сравнении методов по времени работы возникает большее число ситуаций. Отдельно рассмотрим случай обхвата 4 и плотности не менее $2/5$. В таблице 10 указаны отрезки значений плотности, для которых сохраняется соотношение алгоритмов по порядку сложности. Жирным шрифтом выделены названия алгоритмов, «поменявшихся местами» при переходе к очередному отрезку. Для меньших плотностей упорядочение алгоритмов уже описывается общей таблицей 11, в которой значения плотности выражены через обхват. Для компактности

Таблица 10. Расположение алгоритмов по возрастанию оценок времени их работы в случае $g = 4$

$[\rho_1; \rho_2]$	Алгоритмы
$[2/3;\ \ 1]$	фор. 2, фор., «цепи» 2, «сообщения», бэктрэкинг, «цепи»
$[1/2; 2/3]$	фор. 2, **«цепи» 2**, **фор.**, «сообщения», бэктрэкинг, «цепи»
$[2/5; 1/2]$	**«цепи» 2**, **фор. 2**, **«сообщения»**, **фор.**, бэктрэкинг, «цепи»

в первом столбце указаны только правые концы отрезков. Первая строка соответствует случаю наиболее плотных графов с заданным обхватом. Независимо от значения обхвата общая

Таблица 11. Расположение алгоритмов по возрастанию оценок времени их работы в случае $g \geqslant 6$

ρ_2	Алгоритмы
$2/(\ g-2)$	фор. 2, «сообщения», фор., «цепи» 2, бэктрэкинг, «цепи»
$2/(\ g-1)$	фор. 2, «сообщения», **«цепи» 2**, **фор.**, бэктрэкинг, «цепи»
$1/(\ g-3)$	фор. 2, **«цепи» 2**, **«сообщения»**, фор., бэктрэкинг, «цепи»
$1/(\ g-2)$	**«цепи» 2**, **фор. 2**, «сообщения», фор., бэктрэкинг, «цепи»
$2/(2\,g-3)$	«цепи» 2, фор. 2, «сообщения», **бэктрэкинг**, **фор.**, «цепи»
$1/(\ g-1)$	«цепи» 2, фор. 2, «сообщения», бэктрэкинг, **«цепи»**, **фор.**
$1/(2\,g-5)$	«цепи» 2, фор. 2, **бэктрэкинг**, **«сообщения»**, «цепи», фор.
$1/(2\,g-4)$	«цепи» 2, **бэктрэкинг**, **фор. 2**, «сообщения», «цепи», фор.
$1/(2\,g-3)$	«цепи» 2, бэктрэкинг, **«сообщения»**, **«цепи»**, **фор. 2**, фор.
$1/(2\,g-2)$	«цепи» 2, бэктрэкинг, **«цепи»**, **«сообщения»**, фор. 2, фор.

закономерность состоит в том, что для достаточно плотных графов лидируют явные формулы с учётом разреженности, а с уменьшением плотности на первое место выходит второй вариант алгоритма «цепей», обходя матричные методы и алгоритм «передачи сообщений». При дальнейшем понижении плотности эти же алгоритмы обходят и перечислительные методы: бэктрэкинг и первый вариант алгоритма «цепей».

Результаты выполненного сравнения отражают поведение алгоритмов при достаточно больших порядках графов. Кроме того, понятие «плотности» рассмотрено здесь в контексте семейств графов. Для выяснения реальной эффективности методов в случае «недостаточно больших» графов был проведён ряд вычислительных экспериментов.

5. ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ

5.1. Вычислительная среда

Все программы были написаны на языках C и C++ и выполнялись на персональном компьютере семейства x86 с центральным процессором Intel Core 2 Quad Q9650 (3 ГГц, L1 256 КБ, L2 12 МБ), оперативной памятью DIMM DDR2 8 ГБ, 667 МГц, и операционной системой Ubuntu 10.04.

5.2. Программы

Исходный код большинства программ создан автором данной работы. Исключение составляет оригинальная реализация алгоритма «леденцов» [12], незначительно исправленная и дополненная.

В программе на основе явных формул, как и в программе для алгоритма «леденцов», использован класс матриц, входящий в код [12]. Умножение матриц в этом классе осуществляется с помощью процедуры из библиотеки BLAS. В данной работе применялась реализация GotoBLAS2 1.13 [18], технически оказавшаяся наиболее простой в использовании. Умножение матриц с учётом разреженности в явных формулах закодировано отдельно.

Программы для поиска кратчайшего цикла, алгоритма «передачи сообщений» и бэктрэкинга основаны на псевдокоде, предложенном в [16, 9, 10], с учётом всех способов оптимизации, указанных в предыдущем разделе для каждого метода. В алгоритме поиска с возвратом исключены фрагменты, связанные с обработкой флага f («найден хотя бы один цикл, содержащий текущую цепь»), нарушающие правильную работу алгоритма при ограничении длины цикла.

5.3. Графы

С учётом того, что наибольшая возможная плотность неминуемо уменьшается при увеличении обхвата, основной интерес представляет сравнение методов в случаях наиболее плотных графов. Именно с такой целью подобраны первые четыре семейства.

- $K_{n/2;n/2}$ — полные двудольные графы.
- $PG_2(q)$ — графы инцидентности точек и прямых дезарговых конечных проективных плоскостей порядков q, где q — степень простого числа. Граф $PG_2(q)$ содержит $2(q^2+q+1)$ вершин, является регулярным графом степени $q+1$ и имеет обхват 6. В данной работе использовались графы для $q \leqslant 47$. Некоторые из них можно найти в [19]. Каждый граф $PG_2(q)$ имеет наибольший размер и содержит наибольшее количество циклов длиной 6 и, при $q \geqslant 12$, длиной 8 среди всех двудольных графов того же порядка с равномощными долями и обхватом не менее 6 [20].
- $W(q)$ — графы инцидентности точек и прямых некоторых обобщённых четырёхугольников, где q — степень простого числа. Граф $W(q)$ содержит $2\,(q^3+q^2+q+1)$ вершин, является регулярным графом степени q и имеет обхват 8. В данной работе были использованы графы для $q \leqslant 9$, описание которых размещено в [21]. Каждый граф $W(q)$ обладает наибольшим размером и содержит наибольшее количество циклов длиной 8 среди всех двудольных графов того же порядка с равномощными долями и обхватом 8 [20].
- $D(3;q)$ — графы инцидентности точек и прямых флаг-транзитивных полуплоскостей, являющихся проекциями бесконечных полуплоскостей порядков q, где q — степень простого числа [17]. Граф $D(3;q)$ содержит $2\,q^3$ вершин, является регулярным графом степени q и имеет обхват 8. В данной работе привлекались графы $D(3;q)$ для $q=11$ и $q=13$, в дополнение к графам $W(q)$.
- $R_{n/2;n/2}(g;\rho)$ — «случайные» графы с одинаковым количеством вершин в долях, обхватом g и размером, равным ближайшему целому числу к $(n/2)^{1+\rho}$. Рассмотрены случаи $(g;\rho) = (4;3/4), (4;1/2), (4;1/3), (6;1/3)$. Графы $R_{n/2;n/2}(g;\rho)$ генерировались путём последовательного добавления рёбер к пустому графу. Выбор очередного ребра осуществлялся равновероятно из множества ещё не включённых рёбер. В случае $g=6$ рёбра, приводившие к появлению цикла длиной 4, пропускались. Таблица 13 содержит данные о степенях вершин для четырёх «случайных» графов порядка 4500. При одном и том же порядке данные для разных графов почти не отличаются.
- Графы некоторых LDPC-кодов, представленных в [22] (таблица 12). Первые четыре графа использовались в [7], следующие два, а также графы 8000.4000.3.483 и 10000.10000.3.631 — в [9]. Авторы [11] привели результаты работы алгоритма для графа 4095.737.3.101.

Понятие плотности, введённое в разделе 4.5 для семейства графов с неограниченно возрастающим порядком, в случае рассмотренных экстремальных графов удобно «конкретизировать» формально определив $\rho = \log_{n/2} m - 1$. Для каждого полного двудольного графа $K_{n/2;n/2}$, $n > 2$, получаем значение $\rho = 1$, совпадающее со значением «плотности» для семейства. Графам $D(3;q)$ также соответствует точное значение $\rho = 1/3$. В случае графов $PG_2(q)$, начиная с $q = 11$, отличие невелико: $0{,}5 < \rho < 0{,}51$. Для рассмотренных графов $W(q)$ значения

Таблица 12. Характеристики графов некоторых LDPC-кодов [22]

Обозначение графа	n	m	g	ρ
$PEGReg252x504$	756	1 512	8	0,23
$PEGirReg252x504$	756	2 014	6	0,28
816.3.174	1 224	2 448	6	0,22
816.55.178	1 224	4 080	6	0,30
$PEGReg504x1008$	1 512	3 024	8	0,21
$PEGirReg504x1008$	1 512	4 033	6	0,25
4095.737.3.101	4 832	12 285	6	0,21
4095.738.4.102	4 833	16 380	6	0,25
$ram17.5$	7344	14 688	12	0,17
8000.4000.3.483	12 000	24 000	6	0,16
8000.4000.4.484	12 000	32 000	6	0,19
4986.93.128	14 958	34 902	4	0,17
$10002.3.333e$	23 300	39 896	8	0,13
10000.10000.3.631	30 000	60 000	6	0,14
32000.2241.4.106	34 241	128 000	6	0,21

Таблица 13. Статистика степеней вершин в «случайных» графах порядка 4500

g	ρ	$\min d_i$	$\overline{d_i}$	$\max d_i$	$\overline{\lvert d_i - \overline{d_i}\rvert}$
4	3/4	266	326,7	394	12,7
4	1/2	26	47,4	77	4,9
4	1/3	3	13,1	27	2,4
6	1/3	5	13,1	26	1,9

плотности попадают в интервал $(0{,}34; 0{,}41)$. Предложенное определение плотности можно применить и к отдельному графу вне контекста какого-либо семейства, как сделано в таблице 12. Использование такой меры интересно именно с практической точки зрения.

5.4. Корректность вычислений

Правильность работы программ проверялась двумя путями. Во-первых, ввиду большого количества различных алгоритмов, довольно надёжным способом проверки явилось сравнение выходных данных между собой. Во-вторых, для полных двудольных графов и графов инцидентности конечных проективных плоскостей известны явные выражения количества циклов через порядок графа. В случае графов $K_{n_1;n_2}$ можно проверить число циклов произвольной чётной длины k:

$$c_k = \frac{1}{k} \cdot \frac{n_1!}{(n_1 - k/2)!} \cdot \frac{n_2!}{(n_2 - k/2)!}.$$

В случае же графов $PG_2(q)$ выведены формулы для подсчёта циклов длиной 6, 8, 10 и 12 [23]:

$$c_6 = \frac{1}{3}\, a\, q^2,\ c_8 = a\, b,\ c_{10} = \frac{4}{5}\, a\, b\, (q^2 + 1),\ c_{12} = \frac{2}{3}\, a\, b\, (q + 2)(q^3 - 2\, q^2 - q + 3),$$

$$\text{где} \quad a = \binom{q^2 + q + 1}{2}, \quad \text{а} \quad b = \binom{q}{2}^2.$$

Количество коротких циклов в графах $W(q)$, $D(3; q)$ и графах LDPC-кодов, найденное в ходе экспериментов, приводится в приложении.

Все расчёты выполнялись с использованием основных числовых типов данных размером до 64 бит. Для почти всех проведённых экспериментов «запаса» этих типов оказалось достаточно. Однако с увеличением порядка графа количество циклов или промежуточные данные могут не уместиться в отведённый размер. В этом отношении наиболее «экономны» перечислительные

алгоритмы — бэктрэкинг и первый вариант алгоритма «цепей», — для которых переполнение возникает, только если само количество циклов выходит за допустимый диапазон.

Во втором варианте алгоритма «цепей» источником переполнения формально может послужить величина p_{**}, участвующая в сумме (11). Сама сумма никогда не превосходит числа циклов в графе. В остальных методах — алгоритме «передачи сообщений», алгоритме «леденцов» и явных формулах — промежуточные данные в общем случае также оказываются больше числа циклов.

Наиболее «требовательны» к запасу разрядности явные матричные выражения, поскольку в них участвуют количество всех маршрутов заданной длины и близкие по значению величины. Так, при подсчёте циклов длиной 6 в полных двудольных графах с использованием типа double для элементов матриц и векторов переполнение наступает, начиная с графа $K_{1292;1292}$. Однако в случае более разреженных графов $PG_2(q)$ запаса double уже хватило для подсчёта циклов длиной 10. Наибольший граф, участвовавший в экспериментах, имел порядок 4514. При вычислении количества циклов в графах LDPC-кодов можно ограничиться типом float, переход к которому дал двукратное ускорение.

5.5. Результаты

Вычисления выполнялись согласно предположений, принятых в разделе 4.5. Были рассмотрены три значения обхвата: 4, 6 и 8.

При обхвате 4 алгоритм «передачи сообщений» позволяет подсчитывать циклы длиной 4 и 6. В случае плотных графов ($\rho = 1$ и $\rho = 3/4$) соотношение алгоритмов по времени работы, в целом, совпадает с «рейтингом», составленным на основе порядков теоретических оценок. Быстрее всего работают явные формулы, второе место занимают алгоритм «передачи сообщений» и второй вариант алгоритма «цепей», самые медленные — первый вариант алгоритма «цепей» (оказавшийся несколько быстрее) и поиск с возвратом. Соответствующие кривые на рисунке 5 выделены красным цветом. Кривые для явных формул без учёта разреженности не попали на рисунок, поскольку время вычислений составило менее 1 секунды.

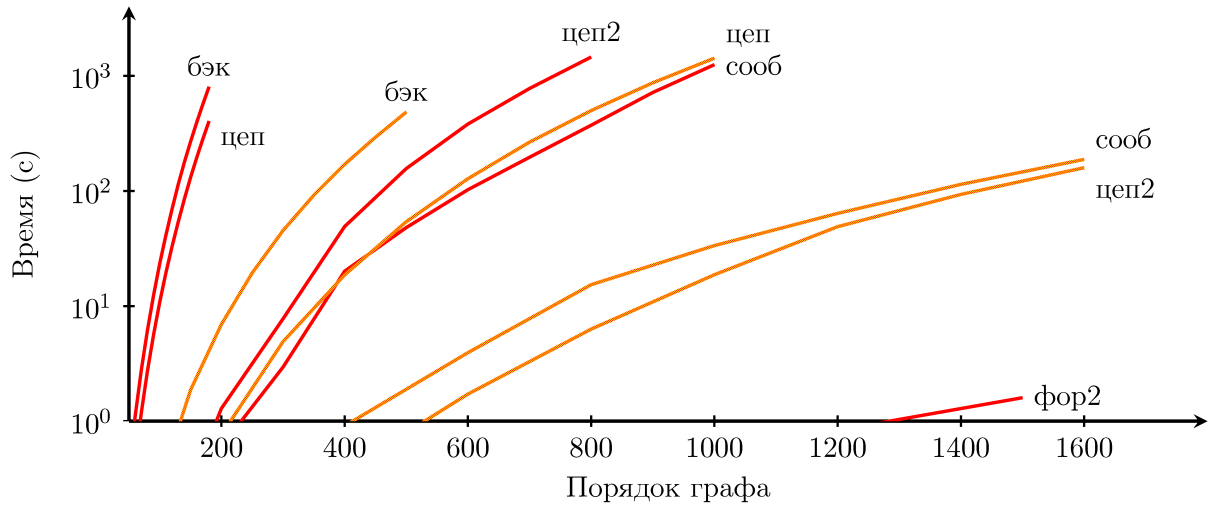

Рис. 5. Время подсчёта циклов длиной 4 и 6 в «плотных» графах с обхватом 4. Цвета соответствуют семействам графов: красный — $K_{n/2;n/2}$, оранжевый — $R_{n/2;n/2}(4;3/4)$.

Для более разреженных графов ($\rho \approx 1/2$, рисунок 6) разница в порядках оценок, принимающих значения с 2,5 по 4, на практике не заметна, хотя отношение временных затрат достигает

нескольких сотен. Аналогичная картина наблюдается для плотности $\rho \approx 1/3$ (графы $W(q)$, $D(3;q)$ и $R_{n/2;n/2}(4;1/3)$), когда порядки оценок ограничены значениями 2 и 3. Кривые не включены в рисунок, так как время расчётов составило несколько секунд.

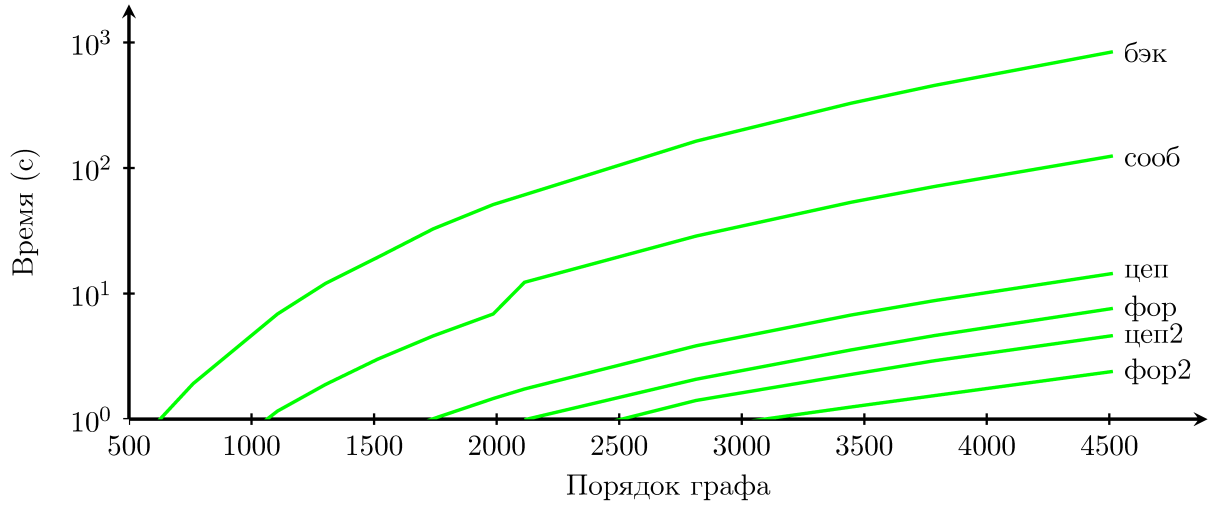

Рис. 6. Время подсчёта циклов длиной 4 и 6 в «разреженных графах»: $PG_2(q)$ (обхват 6) и $R_{n/2;n/2}(4;1/2)$ (обхват 4)

В графах с обхватом 6 подсчитывались циклы длиной 6, 8 и 10. Рисунок 7 содержит кривые для семейства $PG_2(q)$ — зелёные линии, — а также кривые для семейства $R_{n/2;n/2}(6;1/3)$ — синие линии. Кривые, соответствующие явным формулам без учёта разреженности и алгоритму «леденцов», выделены чёрным цветом, так как время вычислений по ним не зависит от плотности графа. В случае наиболее плотных графов ($PG_2(q)$, $\rho \approx 1/2$) соотношение методов по времени работы, в основном, хорошо согласуется с порядками оценок: явные формулы с учётом разреженности — 2,5; алгоритм «передачи сообщений» и явные формулы — 3; второй вариант алгоритма «цепей» — 3,5; поиск с возвратом — 5,5; первый вариант алгоритма «цепей» — 6. Два последних метода на практике «поменялись местами». При меньшей плотности $\rho \approx 1/3$, как и в случае обхвата 4, затруднительно разделить алгоритмы по порядкам роста времени работы. Бэктрэкинг оказался в сотни раз медленнее остальных методов.

Подсчёт циклов длиной 8, 10, 12 и 14 в наиболее плотных графах с обхватом 8 (рисунок 8) также подтверждает распределение алгоритмов по времени работы, составленное на основе оценок их порядков. По причине увеличения длин подсчитываемых циклов ($2\,g - 2$) первый вариант алгоритма цепей и бэктрэкинг могут «обгонять» остальные методы, за исключением второго варианта алгоритма «цепей», только при достаточно низкой плотности графа, причём не абсолютно, а относительно наибольшей достижимой плотности для заданного обхвата (таблица 11).

В завершение раздела рассмотрим соотношение затрат алгоритмов при подсчёте циклов длиной не более $2\,g - 2$ в графах некоторых LDPC-кодов с обхватами 4, 6, 8 и 12 [22]. Результаты выполненных расчётов (таблица 14), в целом, хорошо согласуются с «рейтингом» на основе теоретических оценок (таблица 11), если использовать принятое определение плотности графа. Наиболее существенным отклонением является относительно небольшое время работы первого варианта алгоритма «цепей». В дальнейших комментариях он не рассматривается. В таблице 14 прочерками отмечены случаи, в которых алгоритм «леденцов» и имеющиеся явные формулы не позволяют подсчитать циклы длиной до $2\,g - 2$ включительно. Для графа

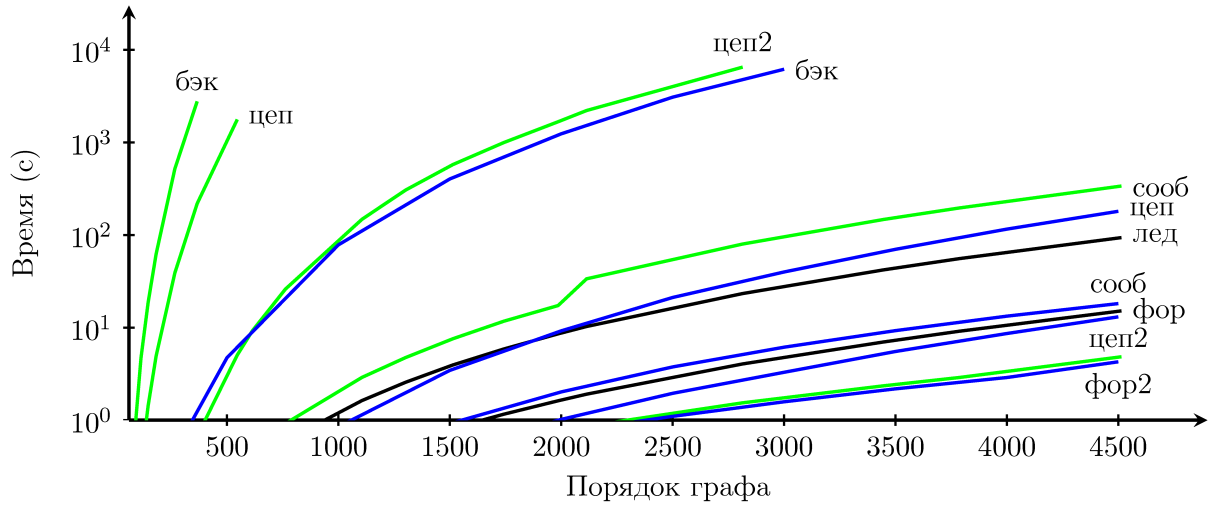

Рис. 7. Время подсчёта циклов длиной 6, 8 и 10 в графах с обхватом 6. Цвета соответствуют семействам графов: зелёный — $PG_2(q)$, синий — $R_{n/2;n/2}(6;1/3)$, чёрный — оба семейства.

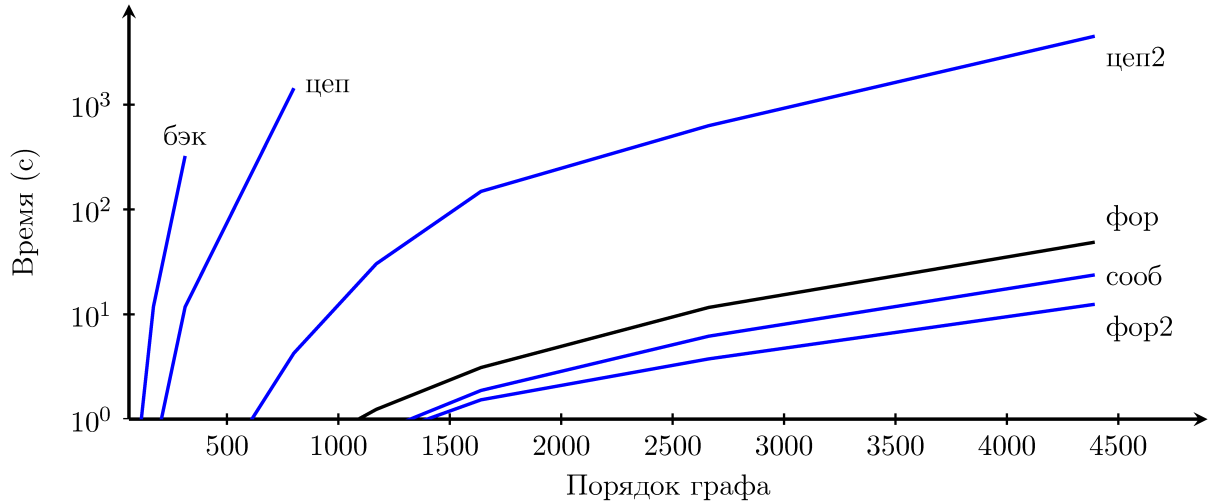

Рис. 8. Время подсчёта циклов длиной 8, 10, 12 и 14 в графах $W(q)$ и $D(3,q)$ (обхват 8)

4986.93.128, наоборот, алгоритм «леденцов» находит количество циклов длиной до $8 = 2\,g$, поэтому время для сравнения также не приводится. Пустые клетки связаны с нехваткой объёма оперативной памяти при использовании матричных методов.

Для первых шести графов ($n \leqslant 1512$) время работы алгоритмов кроме бэктрэкинга оказалось, в основном, менее одной секунды. Выделяются результаты для графа 816.3.174, когда поиск с возвратом опередил матричные методы. По своим характеристикам этот граф как раз соответствует пятой строке таблицы 11. В то же время на графе $PEGReg504x1008$ с близкой плотностью, но бóльшим обхватом, бэктрэкинг существенно отстал. В данном случае для поиска с возвратом порядок примерно равен 3,7, тогда как для других методов, кроме первого варианта алгоритма «цепей», значения находятся в промежутке (2,2; 3]. Аналогичное соотношение получается и в оставшихся четырёх случаях.

Некоторые результаты оказались противоположны выводам на основе теоретических оценок. Например, для графа 816.3.174 можно было предположить, что алгоритм «передачи со-

Таблица 14. Время (с) подсчёта циклов длиной не более $2\,g-2$ для графов LDPC-кодов [22]

	лед	сооб	фор	фор2	цеп	цеп2	бэк
$PEGReg252x504$	–	0,1	0,2	0,2	0,1	0,0	4,1
$PEGirReg252x504$	0,3	0,1	0,1	0,1	0,3	0,1	30
816.3.174	1,0	0,0	0,3	0,2	0,0	0,0	0,1
816.55.178	1,1	0,2	0,3	0,2	0,2	0,1	15
$PEGReg504x1008$	–	0,2	1,2	0,9	0,1	0,1	9,5
$PEGirReg504x1008$	1,9	0,2	0,4	0,4	0,4	0,2	61
4095.737.3.101	42	1,9	5,9	2,9	0,4	0,3	91
4095.738.4.102	42	4,0	6,0	3,0	2,6	1,2	46
$ram17.5$	–	10	–	–	300	17	$35 \cdot 10^4$
8000.4000.3.483	795	1,5	122	26	0,1	0,1	1,5
8000.4000.4.484	800	9,0	122	26	0,4	0,4	24
4986.93.128	–	0,2	116	19	0,0	0,1	0,1
$10002.3.333e$	–	7,6			0,3	0,2	9,6
10000.10000.3.631		5,5			0,2	0,2	3,4
32000.2241.4.106		434			225	84	$31 \cdot 10^4$

общений» и поиск с возвратом уступят явным формулам с учётом разреженности. Такого рода расхождения связаны с недостаточно большими порядками графов и с количественной точки зрения несущественны.

Результаты для графов 4095.737.3.101 и 4095.738.4.102 хорошо согласуются с пятой и четвёртой строками таблицы 11, которым эти графы соответствуют.

Примеры с графами $ram17.5$ ($g = 12$) и 4986.93.128 ($g = 4$) плотностью 0,17 иллюстрируют относительность понятия разреженности при подсчёте циклов длиной $2\,g-2$. Во втором случае плотность графа далека от максимальной и нематричные методы существенно быстрее матричных. Порядки сложности алгоритма «передачи сообщений», второго варианта алгоритма «цепей» и бэктрэкинга примерно равны 2,0, 1,5 и 1,9. В первом же случае плотность графа близка к наибольшей. За счёт возросшей длины подсчитываемых циклов (до 22) порядки сложности этих же трёх методов сильно изменились: 2,3, 2,9 и 4,5.

Оба графа 8000.4000.3.483 и 8000.4000.4.484 с порядками 12000 соответствуют шестой строке таблицы 11 и хорошо отражают влияние плотности на время работы алгоритмов. На этих же примерах заметно сильное отставание явных формул с учётом разреженности, которые согласно теоретических оценок занимают второе место. Вероятно, причиной этому послужил большой объём используемой оперативной памяти, который снизил эффект кэширования.

Граф $10002.3.333e$ — самый разреженный из представленных в таблице 14 ($\rho \approx 0{,}13$). Однако из-за обхвата 8, при котором подсчитываются циклы длиной до 14, перечисление циклов оказалось далеко не самым быстрым способом.

Последние два графа имеют близкие порядки, но существенно отличаются по плотности: 0,14 и 0,21. Порядки сложности алгоритма «передачи сообщений», второго варианта алгоритма «цепей» и бэктрэкинга в первом случае примерно равны 2,3, 1,7 и 2,3, а во втором случае принимают значения 2,4, 2,0 и 2,9. На практике соотношение этих трёх методов по времени работы качественно совпало с теоретическим. Однако переход к большей плотности гораздо сильнее замедлил алгоритмы, чем можно было ожидать исходя из оценок.

5.6. Результаты других работ

Для графов нескольких LDPC-кодов авторы [7, 9, 11] также привели затраты вычислительных ресурсов различных алгоритмов. Указанные в [7, 9, 11] характеристики машин близки к параметрам компьютера, использовавшегося для экспериментов в рамках данной работы.

В сравнении с результатами [7, 9] время расчётов по алгоритму «леденцов», полученное в рамках данной работы, оказалось в 5–6 раз меньше. При этом коэффициент 2 объясняется использованием типа float вместо double.

Реализация алгоритма «передачи сообщений» на графах кодов, использовавшихся в [9], сработала в 15–200 раз быстрее. Основной причиной ускорения явился учёт оптимизаций, предложенных как в [9], так и в данной работе.

Для первого варианта алгоритма «цепей», судя по графу 4095.737.3.101, время вычислений получилось в 70 раз меньше. Использование дополнительного уровня дерева (раздел 4.3) объясняет только четырёхкратное ускорение. Другие факторы, уменьшившие время работы алгоритма, ввиду его недостаточно подробного описания в [11], назвать затруднительно.

5.7. Выводы

В работе рассмотрены пять методов подсчёта циклов в двудольных графах. Алгоритм «леденцов», имеющий сложность того же порядка, что и явные формулы, при обхватах 4, 6 и 8 характеризуется в 7–8 раз (асимптотически — в 4 раза) бóльшим количеством матричных умножений и примерно вдвое бóльшими затратами памяти. Для значений обхвата не менее 10 количество памяти, требуемое алгоритмом «леденцов», постоянно и «достигается» явными формулами только при обхвате 20. Сами наборы выражений для подсчёта циклов длиной g, $g+2$ и $g+4$ выведены в случаях $g = 4, 6, \ldots, 14$.

Для некоторых методов (в данном случае, явных формул и алгоритма «передачи сообщений») обхват графа предполагается известным заранее. Эта особенность не влияет на эффективность метода в целом, так как сложность вычисления обхвата оказывается пренебрежимо мала по сравнению со сложностью подсчёта циклов (по крайней мере, для рассмотренных алгоритмов).

Всеобъемлющее тестирование представляет собой трудоёмкую задачу, так как вычислительные затраты алгоритмов могут определяться разнообразными параметрами: порядком, размером, обхватом, структурой графа, длиной циклов. В данной работе исследованы частные случаи, когда доли графа равномощны, $m \sim (n/2)^2, (n/2)^{7/4}, (n/2)^{3/2}, (n/2)^{4/3}$, $g = 4, 6, 8$ и подсчитываются циклы длиной $g, g+2, \ldots, 2\,g-2$ (по способности алгоритма «передачи сообщений»). Кроме того, в вычислительных экспериментах участвовали графы с порядками до 5000, за исключением семи графов LDPC-кодов. Тем не менее, удалось обнаружить ситуации, в которых более эффективными оказываются различные методы.

В рамках указанной модели явные формулы с учётом разреженности оптимальны для графов с размером $(n/2)^{1+1/(g-2)}$ и более. В случае менее плотных графов эффективнее оказывается второй вариант алгоритма «цепей». Сколько-нибудь существенные затраты памяти достигаются именно для этих двух методов, в остальных случаях гораздо быстрее растёт время работы. Если необходим подсчёт циклов длиной более $2\,g-2$, то из рассмотренных методов теоретически применимы только явные формулы (выведенные для случая $k \leqslant \min\{18; g+10\}$) и перечислительные алгоритмы. При наличии дополнительных особых свойств у исследуемых графов, помимо известного обхвата, возможно дальнейшее упрощение явных выражений.

6. ЗАКЛЮЧЕНИЕ

В работе получены два основных результата. Путём анализа форм замкнутых маршрутов для различных значений длины и обхвата удалось показать, что в случае $k \leqslant 2\,g$ кратности всех сумм в явных формулах понижаются до значений меньше трёх. В результате наиболее трудоёмкой операцией оказывается умножение матриц. Кратность любой тройной суммы можно уменьшить, так как каждый её индекс встречается в парах не более чем с двумя дру-

гими индексами. Вместе с этим обнаружен класс форм, возникающих при $k > 2\,g$, которым соответствуют четырёхкратные суммы.

Исследование нескольких альтернативных алгоритмов подсчёта коротких циклов в двудольных графах позволило выяснить соотношение их вычислительных затрат и сформулировать ряд рекомендаций по применению этих методов в различных ситуациях.

Некоторые задачи, связанные с явными формулами, оказались за пределами выполненной работы.

1. В данном исследовании наборы форм с ограничением на обхват составлялись из имеющихся списков всех двудольных форм (до длины 18). Учёт обхвата при генерации форм, который осуществим, если деревья генерировать отдельно, может позволить продвинуться до бóльших значений длины.
2. Требует дополнительного изучения вопрос о зависимости порядка сложности явных формул от длины цикла и обхвата графа в случае $k > 2\,g$.
3. Представляет интерес разработка процедуры вывода матричных выражений в случае двудольных графов, которая бы оперировала только ненулевыми блоками матриц маршрутов. В результате может сократиться как количество умножений в формулах, так и размер требуемой памяти.
4. Надлежит выяснить, при каких значениях длины цикла и обхвата графа все умножения в формулах содержат в качестве множителя матрицу смежности или её степень. Особенность таких случаев состоит в возможности учесть разреженность графа.
5. Третий способ оптимизации вычислений по явным формулам состоит в упорядочении слагаемых, которое позволило бы как можно раньше уничтожать вспомогательные матрицы.

ПРИЛОЖЕНИЕ

А. КОЛИЧЕСТВО КОРОТКИХ ЦИКЛОВ В ТЕСТОВЫХ ГРАФАХ

Таблица 15. Количество циклов длиной не более $2\,g - 2$ в графах LDPC-кодов

	g	c_g	c_{g+2}	c_{g+4}	c_{g+6}
$PEGReg252x504$	8	802	11 279	86 791	723 426
$PEGirReg252x504$	6	13 244	420 609	13 567 791	
816.3.174	6	132	1 494	9 278	
816.55.178	6	7 921	210 740	6 054 731	
$PEGReg504x1008$	8	2	11 238	91 101	748 343
$PEGirReg504x1008$	6	11 538	408 657	13 110 235	
4095.737.3.101	6	5 183	121 238	3 038 421	
4095.738.4.102	6	43 531	2 052 560	104 144 158	
8000.4000.3.483	6	179	1 218	9 989	
8000.4000.4.484	6	1 620	24 162	408 880	
4986.93.128	4	125	936		
10002.3.333e	8	183	699	3 872	19 488
10000.10000.3.631	6	161	1 260	10 051	
32000.2241.4.106	6	798 345	100 514 280	13 526 254 404	

Для графа $ram17.5$ с обхватом 12 значения c_{12}, c_{14}, …, c_{22} равны

$$154\,224,\ 660\,960,\ 6\,383\,772,\ 53\,878\,848,\ 507\,308\,832,\ 4\,501\,313\,856.$$

Таблица 16. Количество циклов длиной 8, 10, 12 и 14 в графах $W(q)$ и $D(3;q)$

	c_8	c_{10}	c_{12}	c_{14}
$W(2)$	90	72	300	1 080
$W(3)$	1 620	5 184	43 200	336 960
$W(4)$	13 600	97 920	1 387 200	19 094 400
$W(5)$	73 125	936 000	20 280 000	435 240 000
$W(7)$	960 400	27 659 520	1 152 480 000	48 423 916 800
$W(8)$	2 695 680	105 670 656	5 723 827 200	314 068 285 440
$W(9)$	6 725 025	344 321 280	23 528 620 800	1 633 804 473 600
$D(3;11)$	15 007 025	990 902 880	83 385 619 350	7 141 510 846 800
$D(3;13)$	62 719 956	6 152 724 864	743 220 511 176	91 700 663 779 296

СПИСОК ЛИТЕРАТУРЫ

1. Harary F., Manvel B. On the Number of Cycles in a Graph. *Matematický časopis*, 1971, vol. 21, no. 1, pp. 55–63.
2. Chang Y.C., Fu H.L. The Number of 6-Cycles in a Graph. *The Bulletin of the Institute of Combinatorics and Its Applications*, 2003, vol. 39, pp. 27–30.
3. Perepechko S.N., Voropaev A.N. The Number of Fixed Length Cycles in an Undirected Graph. Explicit Formulae in Case of Small Lengths. *International Conference «Mathematical Modeling and Computational Physics» (MMCP'2009)*. Dubna : JINR, 2009, pp. 148–149.
4. Воропаев А.Н., Перепечко С.Н. Количество простых циклов фиксированной длины в неориентированном графе. Явные формулы в случае малых длин. *Письма в журнал «Физика элементарных частиц и атомного ядра»*, принято к публикации.
5. Воропаев А.Н. Вывод явных формул для подсчёта циклов фиксированной длины в неориентированных графах. *Информационные процессы*, 2011, том 11, № 1, стр. 90–113. `http://www.jip.ru/2011/90-113-2011.pdf`.
6. Flum J., Grohe M. The Parameterized Complexity of Counting Problems. *SIAM Journal on Computing*, 2004, vol. 33, no. 4, pp. 892–922.
7. Halford T.R., Chugg K.M. An Algorithm for Counting Short Cycles in Bipartite Graphs. *IEEE Transactions on Information Theory*, 2006, vol. 52, no. 1, pp. 287–292.
8. Воропаев А.Н., Перепечко С.Н. Явные формулы для подсчёта простых циклов с длиной, близкой к обхвату графа. *XVII международная научная конференция «Математика. Компьютер. Образование» (МКО-2010)*. Ижевск : РХД, 2010, выпуск 17, стр. 16.
9. Karimi M., Banihashemi A.H. A Message-Passing Algorithm for Counting Short Cycles in a Graph. *2010 IEEE Information Theory Workshop (ITW 2010, Cairo)*. 2010, pp. 1–5. Полная версия — `http://arxiv.org/abs/1004.3966`.
10. Tarjan R. Enumeration of the Elementary Circuits of a Directed Graph. *SIAM Journal on Computing*, 1973, vol. 2, no. 3, pp. 211–216.
11. Nguyen D.V., Chilappagari S.K., Marcellin M.W., Vasic B.V. LDPC Codes from Latin Squares Free of Small Trapping Sets. *CoRR*, 2010, vol. abs/1008.4177, `http://arxiv.org/abs/1008.4177`.
12. USC — Keith M. Chugg. `http://csi.usc.edu/chugg/tools/cycles.shtml`.
13. Харари Ф. *Теория графов*. М. : Мир, 1973. (Harary F. *Graph Theory*. Reading : Addison-Wesley, 1969.)
14. Explicit Formulae : FlowProblem. `http://flowproblem.ru/cycles/explicit-formulae`.
15. Караваев А.М., Воропаев А.Н. Эффективность распараллеливания явных формул для подсчёта коротких циклов в графе. *Международная научная конференция «Параллельные вычислительные технологии» (ПаВТ'2010)*. Челябинск : Изд. центр ЮУрГУ, 2010, стр. 486–497.
16. Itai A., Rodeh M. Finding a Minimum Circuit in a Graph. *SIAM Journal on Computing*, 1978, vol. 7, no. 4, pp. 413–423.

17. Lazebnik F., Ustimenko V.A., Woldar A.J. A New Series of Dense Graphs of High Girth. *Bulletin (New Series) of the American Mathematical Society*, 1995, vol. 32, no. 1, pp. 73–79.

18. Texas Advanced Computing Center: GotoBLAS2. `http://www.tacc.utexas.edu/tacc-projects/gotoblas2`.

19. Projective Planes of Small Order. `http://www.uwyo.edu/moorhouse/pub/planes`.

20. De Winter S., Lazebnik F., Verstraëte J. An Extremal Characterization of Projective Planes. *The Electronic Journal of Combinatorics*, 2008, vol. 15, no. R143, pp. 1–13.

21. Generalised Polygons of Small Order. `http://www.uwyo.edu/moorhouse/pub/genpoly`.

22. Encyclopedia of Sparse Graph Codes. `http://www.inference.phy.cam.ac.uk/mackay/codes/data.html`.

23. Lazebnik F., Mellinger K.E., Vega O. On the Number of k-gons in Finite Projective Planes. *Note di Matematica*, 2009, vol. 29, no. 1, pp. 135–152.