Московский государственный университет имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра информационной безопасности

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Разработка итерационных алгоритмов поиска автоморфизмов и изоморфизмов комбинаторных объектов.

Автор:
группа 419
Ефремов Степан Сергеевич

Научный руководитель:
доцент, к.ф.- м.н.
Егоров Владимир Николаевич

Москва, 2018

Содержание

1 Введение

Проблема изоморфизма графов в настоящее время приобрела статус одной из самых сложных и важных с теоретической точки зрения задач комбинаторной математики. Известно, что решение задачи нахождения изоморфного подграфа с точки зрения алгоритмической сложности позволило бы ответить на вопрос о совпадении или различии классов P и NP. В последние годы сформировались два направления изучения и решения данной проблемы. Первое направление - теоретическое, в котором проблема изоморфизма рассматривается с позиций соременной теории сложности алгоритмов и вычислений, а второе - сугубо практическое, предполагающее разработку алгоритмов, решающих задачу изоморфизма графов за "практически приемлемое"время.

Первые серьёзные результаты в теоретическом направлении появились в 50-х и 60-х годах прошлого века. Для нескольких семейств графов была установлена полиномиальная сложность нахождения изоморфизма. В 70-х годах, особенно после выхода в свет замечательной монографии Нормана Биггса "Алгебраическая теория графов к решению этой задачи подключились крупные специалисты в области линейной алгебры, теории групп и ряда других направлений общей алгебры. Результаты не замедлили сказаться - открыто много новых интересных алгебраических свойств графов. В 1979 году Егорову В.Н. удалось получить положительное решение гипотезы Адама об изоморфизме графов с циркулянтными матрицами смежности вершин порядка свободного от квадратов [1]. В дипломной работе рассматривается итерационный алгоритм поиска автоморфизмов и изоморфизмов графов.

Поиск автоморфизмов и изоморфизмов алгебраических систем, в том числе графов, является известной задачей. Опубликовано множество книг и научных статей, посвященных данной проблеме. Из существующих алгоритмов на данный момент стоит выделить "кратко об алг. Лакса (про ограниченную Валентность)", "кратко об алг. в случае циркулянтной матрицы смежности".

Как выяснилось, эффективного (полиномиального) алгоритма, который был бы универсальным, т.е. применим к любым графам, сейчас не существует. На основании этого было решено исследовать итерационный алгоритм, предложенный в статье В.Н.Егорова [1].

Работа разбита на следующие пункты:

1. Основные определения и обозначения: приводятся основные понятия, используемые в ходе работы.

2. Исследование и модернизация алгоритма: в данном пункте описаны полученные оценки сложности алгоритма, а также способы модернизации, которые удалось найти в ходе исследования. Особое внимание уделяется исследованию расспараллеливания вычислений программной реализации.

3. Практическое применение: рассказывается о различных применениях алгоритма для графов и некоторых других комбинаторных объектов. В том числе, рассматриваются способы использования алгоритма для задачи Коши.

4. Разработка программных модулей: рассматриваются ключевые моменты, на которые стоило уделить особое внимание при реализации програмных продуктов. В результате получены 2 программы: упрощенный вариант с графическим интерфейсом для использования на гра-

фах с небольшим количеством вершин; вариант реализации со всеми модернизациями, упомянутыми во 2-ом пункте, для более удобного сбора информации и исследования эффективности.

5. Результаты опробывания программ: в этом пункте приводится анализ результатов, полученных при тестировании разработанных программ на различных данных, также подробно проанализированы результаты запусков программы на суперкомьпьютере Ломоносов.

2 Постановка задачи

Целью курсовой работы является модернизация итерационного алгоритма поиска автоморфизмов графа, предложенного в статье В.Н.Егорова [1] и его реализация в виде программы, поддерживающей графичесий интерфейс для удобного ввода графа и получения итоговых и промежуточных результатов. А также исследование времени работы полученной программы на различных размерах входных данных и определение ограничения на максимально возможное количество вершин вводимого графа.

Автоморфизм графа - есть отображение множества вершин на себя, сохраняющее смежность. Множество таких автоморфизмов образует вершинную группу графа или просто группу графа [2]. Ставится задача поиска таких групп.

Поставленная задача не является простой, особенно для графов, имеющих большое количество вершин. Для ее решения граф удобно представить в виде матрицы смежности. Пусть, для удобства, элементы этой матрицы состоят из единиц и нулей. Если на пересечении строки i и столбца j стоит 1(0), и $i \neq j$ - это означает, что существует (не существует) ребро (i,j), если $i = j$, вершина под номером i имеет (не имеет) петлю. При отображении одной вершины в другую, в матрице смежности меняются местами строки и столбцы соответствующих вершин. Автоморфизмом является следующее отображение: $\widehat{g}A\widehat{g}^{-1} = A$, где $\widehat{g}$ - матричный вид подстановки g. Польза такого представления графа заключается в том, что для поиска автоморфизма g, являющегося отображением множества вершин графа G на себя, достаточно проверить равенство элементов матрицы смежности до и после применения подстановки g.

3 Описание алгоритма

Идея алгоритма заключается в построении последовательности множеств частичных отображений по матрице смежности графа. Последнее множество из этой последовательности будет содержать в себе все автоморфизмы графа.

Для описания построения такой последовательности будут использоваться следующие определения и обозначения.

3.1 Определения и обозначения

$A = A^{n\times n}$ - матрица смежности графа $G(V, E)$ размера $n \times n$; a_{ij} - элементы матрицы , $i, j = 1, \ldots, n = |V|$.

Частичным отображением g_k^s (s - обозначает индекс элемента в множестве M_k) называется подстановка вида

$$\begin{pmatrix} 1 & 2 & \ldots & k & k+1 & \ldots & n \\ r_1 & r_2 & \ldots & r_k & q_{k+1} & \ldots & q_n \end{pmatrix}$$

, где $r_1, \ldots, r_k$ - фиксированные (заданные) элементы, $q_{k+1}, \ldots, q_n$ - произвольные, причем последовательность $r_1, \ldots, r_k$, $q_{k+1}, \ldots, q_n$ является перестановкой вершин заданного графа.

M_k - множество, состоящее из элементов g_k^s, $s = 1, \ldots, n_k = |M_k|$.

$|M_k|$ - количество элементов g_k^s в множестве.

M'_k - множество элементов, которые содержатся в M_k и удовлетворяют критерию h.

h - критерий (описанный в разделе 2.2).

$\{M'_k\}$ - последовательность множеств M'_k, $k = 1, \ldots, n$.

3.2 Проверка по критерию

Критерием h является проверка подматриц на равенство. Предположим, требуется определить, удовлетворяет ли элемент g_k^s критерию h. Для этого необходимо, чтобы элементы a_{ij}, для всех $i, j \leq k$, были равны соответствующим элементам $a_{r_i r_j}$. Подматрица g_k^s выглядит так:

$$\begin{array}{c c} & \begin{array}{cccc} r_1 & r_2 & \dots & r_k \end{array} \\ \begin{array}{c} r_1 \\ r_2 \\ \vdots \\ r_k \end{array} & \begin{pmatrix} a_{r_1 r_1} & a_{r_1 r_2} & \dots & a_{r_1 r_k} \\ a_{r_2 r_1} & a'_{r_2 r_2} & \dots & a_{r_2 r_k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r_k r_1} & a_{r_k r_2} & \dots & a_{r_k r_k} \end{pmatrix} \end{array}$$

Если хотя бы одно из равенств не выполнено, это означает, что по данной частичной подстановке хотя бы одна вершина отобразилась в другую так, что структура графа изменилась. А так как частичная перестановка g_k^s фиксирует $r_1, \dots r_k$, то структура останется измененой, что означает отображение не будет являться автоморфизмом.

3.3 Построение множеств частичных отображений

Описание построения M'_n.

На первом этапе рассматриваются все g_1^s, образующие множество M_1. Каждый элемент g_1^s проверяется по критерию h. Элементы, удовлетворяющие h, образуют M'_1. В каждый элемент из M'_1 добавляется еще одно отображение $(2 \to r_2)$, т.о. происходит переход от g_1^s к g_2^s. Из каждого g_1^s получается разных $(n-1)$ элементов g_2^s. Всевозможные элементы g_2^s образуют M_2. Далее строится M'_2, добавляя в него только те элементы из M_2, которые удовлетворяют критерию. Аналогично получаются множества

$M_3,\ M_3',\ M_4, \ldots, M_n,\ M_n'$.

Таким образом, получается последовательнось множеств частичных отображенией:

$\{M_k'\}$: $M_1' \subseteq M_2' \subseteq \ldots \subseteq M_n'$.

Множество M_n' является множеством всех возможных автоморфизмов (или пустое, если их нет).

4 Оценка сложности алгоритма

Алгоритм применим для любых графов, но для удобство рассматриваются случайные графы, матрицы смежности которых состоят из нулей и едениц. В этом случае получена оценка сложности алгоритма в среднем.

Так как граф случайный, элементы матрицы смежности графа равны 1 или 0 с вероятностью $\frac{1}{2}$. Необходимо вычислить наиболее вероятные размеры для каждого множества M'_k.

Если рассмотреть построение множеств, не учитывая промежуточного критерия, то на k-ом шаге множество увеличивается в $(n-k)$ раз (так как для каждого элемента $g^s_{(k-1)} = (r_1, \dots, r_{k-1})$ добавляется r_k из оставшихся $(n-k)$ вариантов): $|M_{k+1}| = |M_k|(n-k) = n(n-1)\dots(n-k)$.

С учетом критерия получается:

1. Из построения множества M'_1 следует $|M'_1| \approx \frac{|M_1|}{2}$ (элемент $a_{11} = a_{r_1 r_1}$ с вероятностью $\frac{1}{2}$).

2. На $(k+1)$-ом шаге требуется совпадение $((k+1)+(k+1)-1)$ элементов. Так как граф случайный (матрица с равновероятным распределением 0 и 1), то на этом шаге $|M'_{k+1}| = \frac{1}{2^{2m+1}}|M_k|$ элементов.

Учитывая построение $\{M'_k\}$ и пункты 1, 2 получим

$|M'_{k+1}| \approx \frac{n!}{(n-k-1)!\ 2^{(k+1)^2}} \approx \frac{1}{e^{k+1}} \frac{n^{n+1/2}}{(n-k-1)^{(n-k-1/2)}\ 2^{(k+1)^2}}$. Последнее равенство получено используя формулу Стирлинга: $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$, при большом n.

Далее необходимо рассмотреть последовательность $\{|M'_{k+1}|\}$.

$|M'_{k+1}| = \frac{n(n-1)\dots(n-k)}{2^{(k+1)^2}} \leq \frac{n^{k+1}}{2^{(k+1)^2}}$

Равенство $n^{k+1} = 2^{(k+1)^2}$ означает, что в множестве осталось небольшое количество элементов. Данное равенство далее будет называться стабилизацией последовательности $\{M'_k\}$, а k - значением стабилизации.

После решения данного равенства, получается, что стабилизация наступает при $k \approx \log_2(n)$. Это означает, что с вероятностью близкой к единице (для случайного графа) уже на $k = \lceil \log_2(n) \rceil$ шаге мы обнаружим отсутствие или наличие автоморфизма (в статье это выдвигается как тезис).

Вычисление номера множества k в последовательности $\{|M'_k|\}$, соответствующий самому большому множеству:

Пусть $|M'_{k+1}| : |M'_{k+1}| = f(k+1)$,

$f'(k+1) = \frac{\ln(n) n^{k+1} 2^{(k+1)^2} - (2k+2)\ln(2) 2^{(k+1)^2} n^{k+1}}{(2^{(m+1)^2})^2}$.

Получается уравнение:

$\ln(n) - (2k+2)\ln(2) = 0 \;\Rightarrow\; k+1 \approx \frac{5}{7}\ln(n) - 1$.

Так как k и n целые, то

$k+1 = \lceil (\frac{5}{7}\ln(n) - 1) \rceil)$.

Необходимо отметить, что в тот момент, когда $k \approx \log_2(n)$, можно судить о том, существуют автоморфизмы в графе или нет. Если размер множества $M_k = 1$ (только тождественная подстановка), можно утверждать с вероятностью близкой к 1, что автоморфизмов, кроме тривиальной подстановки, не существует. Если $M_k > 1$, то, наоборот, с вероятностью близкой к 1 автоморфизм существует. Данный факт позволяет сэкономить память и уменьшить количество операций при поиске автоморфизмов в тех графах, в которых они существуют.

Опираясь на вышесказанное, получены оценки:

- значение стабилизации

  $k \approx \log_2(n)$

- номер наибольшего по количеству элементов множества

  $k \approx \frac{5}{7}\ln(n)$

- ограничение на размер множества

  $|M'_k| = \frac{n(n-1)...(n-k-1)}{2^{(k)^2}} \leq \frac{n^k}{2^{(k)^2}}$

- приблизительное количество операций в секунду:

  $\frac{n^{(5/7)\ln(n)}}{2^{((5/7)\ln(n))^2}} \times n(2(\frac{5}{7}\ln(n)) + 1) \times \log_2(n)$

- затраты оперативной памяти:

  $2 \times \frac{n^{(5/7)\ln(n)}}{2^{((5/7)\ln(n))^2}} \times \log_2(n)$

5 Модернизация

На основе результатов тестирования программы и анализа выяснилось, что эффективность алгоритма тесно связана с тем, как нумеруются вершины графа. Другими словами, время работы программы зависит от того, какой матрицей смежности (из многих) представляется граф.

Модернизация заключается в том, что на каждом этапе можно требовать, чтобы мощность множества частичных отображений была минимальна. Однако представить матрицу смежности нужным образом не представляется возможным при больших размерах, и время работы, затрачеваемой на это, превышает время работы алгоритма. Например, при $n = 100$ потребуется всего лишь 100 операций, чтобы выяснить, с какой вершины эффективнее всего начать отсчет на первой итерации. Но для того, чтобы получить выгоду на второй итерации, уже требуется более 10 000 операций [3]. Поэтому оптимальным является уменьшить только начальное множество частичных отображений.

Для получения первого множества, мощность которого будет наименьшей, в изначальной матрице меняется порядок строк/столбцов (переименование вершин), а именно, необходимо поменять строки и соответствующие им столбцы таким образом, чтобы на месте первого элемента главной диагонали стояло то значение, которое встречается меньше всех других на диагонали. То есть, если на главной диагонали 70% нулей и 30% единиц, необходимо поместить на первую позицию единицу. Эффективность данной модификации исходит из того, что первая итерация алгоритма составляет множество из тех номеров строк (столбцов), в которых значение на главной диагонали совпадает с первым элементом главной диагонали.

Значит, выбрав на эту позицию наименьший по количеству встречаний на диагонали элемент, получается наименьшее по мощности множество.

Оценки получены для случайных матриц (вероятность нуля и единицы в каждой позиции одинакова и равна $\frac{1}{2}$). Предположим, что в матрице $n \times n$ на главной диагонали находится k нулей, где $k \leq \frac{1}{2}n$ (если количество нулей больше половины, то за k обозначается количество единиц). Тогда, если на первое место главной диагонали выбирать элемент случайным образом, получим, что математическое ожидание размера полученного множества будет $\frac{k}{n} * k + \frac{n-k}{n} * (n-k)$. В модифицированном алгоритме всегда будет получаться k. Таким образом, улучшение составляет $\frac{\frac{k}{n}*k+\frac{(n-k)}{n}*(n-k)}{k} = 2\frac{k}{n} + \frac{n}{k} - 2$ раз. В случае диагонали, состоящей из одних нулей (единиц), то есть k = 0, улучшение равняется 1 (мнощьность множеств одинакова). Для получения полной оценки необходимо посчитать матожидание улучшения для произвольного числа нулей и единиц на главной диагонали. Вероятноть k нулей составляет $\frac{C_n^k}{2^n}$. Тогда матожидание улучшения $\frac{1}{2^{n-1}} + 2\sum_{i=1}^{\lceil \frac{n}{2} \rceil} \frac{C_n^k}{2^n} * (2\frac{k}{n} + \frac{n}{k} - 2)$. Далее в таблице приведены значения для различных n:

количество вершин графа	коэф. уменьшения мощности
4	2.12500
8	2.11198
14	1.69565
20	1.51723
50	1.27697
100	1.18312
200	1.12400

6 Результаты

Получена приблизительная сложность работы усовершенствованного алгоритма и в случае случайного графа, модернизация алгоритма дает неплохой результат на небольших размерах. Но эффективность стремительно падает, при увеличении количества вершин графа.

Время работы программы в зависимости от размера графа на среднем по мощности компьютере (затраты оперативной памяти $\approx 2 \times 10^9$ байт):

1. До 200 вершин - < 1 секунды.
2. В пределах 300 вершин - около 2 секунд.
3. 400 — 500 вершин - до 20 секунд.

Приблизительная оценка времени работы программы на мощной вычислительной технике: оперативная память 10^{14} байт, количество операций в секунду 10^{15} [5].

1. 1000 вершин - < 1 секунды (10^{12} операций).
2. 3000 вершин - < 5 секунд (5×10^{15} операций).
3. 10000 вершин - около $2,5$ часов (10^{19} операций).
4. 100000 вершин - около 320000 лет (10^{28} операций).

В последнем случае требуется 10^{21} байт оперативной памяти.

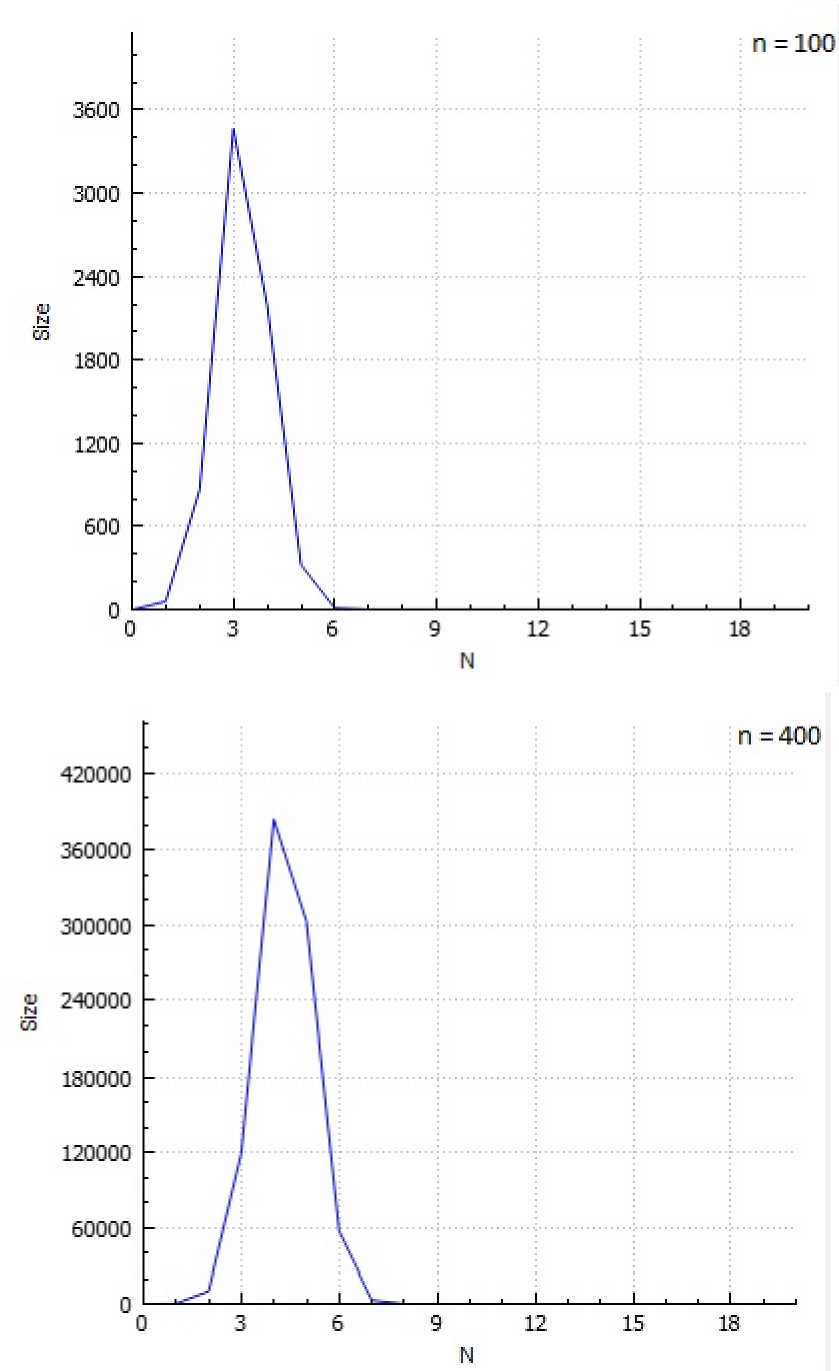

Рис. 1: Построение $\{|M_m|\}$

Заключение

В результате данной работы разработан модернезированный итерационный алгоритм поиска автоморфизмов и изоморфизмов графов, основанный на алгоритме, предложеном в статье [1]. Получены оценки сложности алгоритма при больших размерах графа (количество вершин $n > 1000$).

Реализована программа, решающая поставленную задачу на персональном компьютере, при $n \leq 500$, а также вычислены значения затрачиваемых ресурсов, при работе программы на суперкомпьютере (при $n \leq 10^4$).

Список литературы

[1] *В.Н. Егоров, А.В. Егоров.* Группы автоморфизмов и изоморфизм комбинаторных объектов и алгебраических структур // *Ломоносовские чтения, Научная конференция, Москва, факультет ВМК МГУ имени М.В.Ломоносова.*

[2] *В.Н. Егоров.* О группах автоморфизмов матриц // *Прикладная дискретная математика.*

[3] Алгоритмы: построение и анализ. / Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн.

[4] *В.В Воеводин.* Математические основы параллельных вычислений. — Изд. Моск. ун-та, 1991.

[5] *В.В. Воеводин, Вл.В. Воеводин.* Параллельные вычисления. — БХВ - Петербург, 2002.

[6] Практика суперкомпьютера «Ломоносов» / Воеводин Вл., Жуматий С., Соболев С. et al. // *Открытые системы.* — 2012.