МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Распределённая система вычисления инвариантов графов

АВТОРЕФЕРАТ

дипломной работы

студента 6 курса 631 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Томилова Дмитрия Александровича

Научный руководитель

д. ф.-м. н., доцент _____ М. Б. Абросимов

22.01.2022 г.

Заведующий кафедрой

д. ф.-м. н., доцент _____ М. Б. Абросимов

22.01.2022 г.

Саратов 2022

ВВЕДЕНИЕ

В современном мире множество задач требует затратных вычислений и обработки большого количества данных. Такие задачи встречаются в разных областях науки, например, математике, молекулярной биологии, медицине, астрофизике, в том числе и в теории графов. В связи с этим появилось понятие распределенных вычислений – способа решения трудоемких задач с использованием нескольких вычислительных узлов – ядер или целых компьютеров. Существуют различные виды распределенных вычислений – кластерные, облачные, добровольные и другие.

Добровольные вычисления – это тип распределенных вычислений, при котором люди жертвуют неиспользованные ресурсы своих компьютеров на исследовательский проект. Фундаментальная идея, лежащая в основе этого, заключается в том, что современный компьютер достаточно мощный, чтобы выполнять миллиарды операций в секунду, но для большинства пользователей используется только 10-15% его мощности. Типичное использование, такое как базовая обработка текста или просмотр веб-страниц, по большей части оставляет компьютер без дела. С другой стороны, некоторые научные исследования не могут обойтись без помощи сложных компьютерных вычислений. Именно для этих целей создается проект – распределенная система вычислений инвариантов графов.

В данной работе использование добровольных вычислений будет рассмотрено в качестве механизма решения некоторых сложных задач из теории графов на примере вычисления кликового числа, хроматического индекса и индекса палитры.

Дипломная работа состоит из введения, 2 разделов, заключения, списка использованных источников и 8 приложений. Общий объем работы – 70 страниц, из них 34 страниц – основное содержание, включая 18 рисунков и 6 таблиц, список использованных источников из 16 наименований.

КРАТКОЕ СОДЕРЖАНИЕ

1 Теоретические сведения

В данном разделе описываются основные понятия для ознакомления с графами и добровольными вычислениями.

1.1 Основные определения

В данном разделе описываются основные определения и теоретические положения, необходимые для понимания предметной темы.

Неориентированным графом (далее просто графом) называется пара $G = (V, \alpha)$, где α – симметричное и антирефлексивное отношение на множестве вершин V, называемое отношением смежности.

Если $(u, v) \in \alpha$, то говорят, что вершины u и v *смежны* и эти вершины соединены ребром (u, v). При этом (u, v) и (v, u)это одно и то же ребро, которое обозначают $\{u, v\}$.

Путем в графе $G = (V, \alpha)$ называется последовательность вершин и ребер вида $v_0, \{v_0, v_1\}, v_1, \dots, \{v_{n-1}, v_n\}, v_n$. При этом говорят, что путь соединяет вершины v_0 и v_n и вершина v_n *достижима* из v_0.

Будем считать, что каждая вершина достижима из самой себя. Тогда отношение достижимости является отношением эквивалентности на множестве вершин графа. Классы этого отношения называются *компонентами связности* графа. Граф с универсальным отношением достижимости называется *связным*.

Полным графом называется граф, в котором каждая пара различных вершин смежна. Полный граф с n вершинами обозначается K_n.

Степенью вершины v в графе G называется количество вершин в G, смежных с v. Набор чисел, являющихся степенями вершин графа G, называется его *степенным множеством*, а вектор, составленный из степеней вершин графа G в порядке невозрастания, – *вектором степеней*. *Максимальная степень* обозначается Δ.

Кликой графа $G = (V, \alpha)$, называется подмножество его вершин $C \in V$, любые две из которых соединены ребром. Другими словами, клика графа G – полный подграф графа G.

Максимальная клика – это клика, которая не может быть расширена путём включения дополнительных смежных вершин, то есть нет клики большего размера, включающей все вершины данной клики.

Наибольшая клика – это клика максимального размера для данного графа.

Кликовое число – это число вершин в наибольшей клике графа G.

Кубическим графом называется граф, все вершины которого имеют степень 3.

Паросочитанием называется множество попарно несмежных ребер, то есть ребер.

Совершенным паросочетанием называется паросочетание, в котором участвуют все вершины графа, т.е. любая вершина графа инцидентна ровно одному ребру, входящему в паросочетание.

Вершинной раскраской графа $G(V, \alpha)$ называется отображение φ из множества вершин V во множество красок $\{c_1, \dots, c_t\}$, что для любых двух различных смежных вершин v_1, v_2 верно $\varphi(v_i) \neq \varphi(v_j)$.

Хроматическим числом $\chi(G)$ графа $G(V, \alpha)$ называется такое минимальное число t, что существует вершинная раскраска графа в t цветов.

Реберной раскраской графа $G(V, \alpha)$ называется отображение φ из множества ребер α во множество красок $\{c_1, \dots, c_t\}$, что для любых двух различных ребер α_i, α_j, инцидентных одной вершине, верно $\varphi(\alpha_i) \neq \varphi(\alpha_j)$.

Хроматическим индексом $\chi'(G)$ графа $G(V, \alpha)$ называется такое минимальное число t, что существует реберная раскраска графа в t цветов.

Имеет место *теорема Визинга*, которая гласит, что если $G(V, \alpha)$ – простой граф, то либо $\chi'(G) = \Delta$, либо $\chi'(G) = \Delta + 1$. При этом графы, для которых $\chi'(G) = \Delta$, называются графами I типа, а для которых $\chi'(G) = \Delta + 1$ –

графами II типа. Задача нахождения хроматического индекса графа является NP-полной.

Палитрой вершины называется множество цветов ребер, смежных с ней.

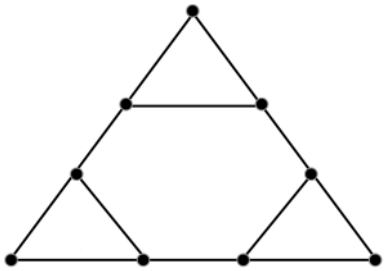
Индексом палитры графа называется минимальное число разных палитр вершин и обозначается $\check{s}(G)$.

Известны следующие значения индекса палитры графов:

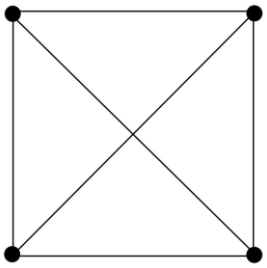
1. Индекс палитры графа G равен 1 тогда и только тогда, когда граф G – регулярный граф I типа.
2. $\check{s}(K_n) = \begin{cases} 1, & \text{если } n \equiv 0 \pmod 2 \text{ или } n = 1 \\ 3, & \text{если } n \equiv 3 \pmod 4 \\ 4, & \text{если } n \equiv 1 \pmod 4 \end{cases}$
3. Для кубических графов верно:

$$\check{s}(G) = \begin{cases} 1, & \text{если } G - \text{граф I типа} \\ 3, & \text{если } G - \text{граф II типа с совершенным паросочетанием} \\ 4, & \text{если } G - \text{граф II типа без совершенного паросочетания} \end{cases}$$

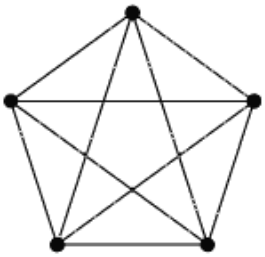
Некоторые известные значения индекса палитры, вычисленные по приведенным выше формулам, приведены на рисунке 1.



Граф Серпинского S_3^2 с $\check{s} = 3$



Граф K_4 с $\check{s} = 1$



Граф K_5 с $\check{s} = 4$

Рисунок 1 – Некоторые графы и их индекс палитры

1.2 Алгоритм вычисления хроматического индекса

В данном разделе описывается алгоритм нахождения хроматического индекса графов.

1.3 Алгоритм вычисления индекса палитры

В данном разделе описывается алгоритм нахождения индекса палитры графов.

1.4 Алгоритм нахождения кликового числа графа

В данном разделе описывается алгоритм нахождения кликового числа графов.

2.1 Общие сведения о системе

Вычисление различных инвариантов графов, для количества вершин больших 10, может работать достаточно долгое время при использовании обычных персональных компьютеров. Поэтому система позволяет распределить нагрузку между подключаемыми компьютерами в локальной сети или в сети интернет. На компьютере-сервере запускается серверное приложение, указывается инвариант, который нужно считать. Любой пользователь на своем компьютере запускает программу-клиент может подключаться к серверу и начать считать присылаемые на выполнение задачи. Причем не обязательно обновлять клиентское приложение для получения информации об алгоритме подсчета запущенного на сервере инварианта: загрузка необходимых способов происходит в процессе выполнения программы-клиента. Схема модулей системы, приведена на рисунке 2. Тонкими стрелками указана зависимость одних модулей от других. Двойными стрелками указано взаимодействие между модулями на основе удаленного вызова процедур.

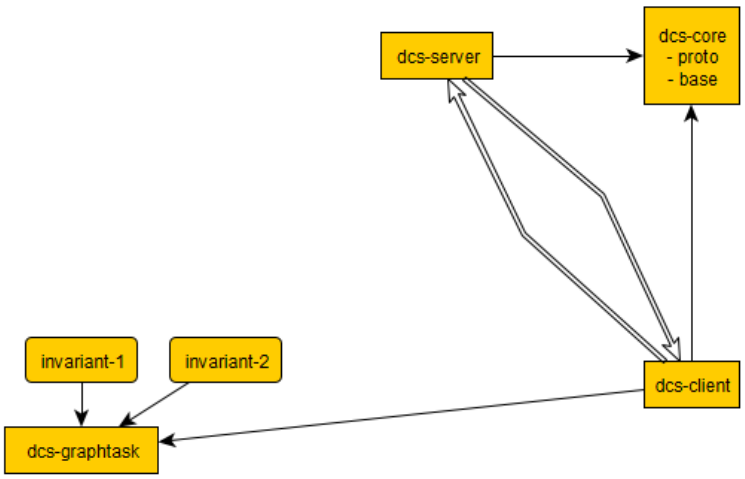


Рисунок 2 – Архитектура системы

2.2 Используемые технологии

В данном разделе описываются технологии, которые были использованы для реализации системы распределенных вычислений.

Для работы с графами существуют раличные пакеты, в частности, пакет утилит Nauty, который содержит различные генераторы, которые порождают разные классы графов, например, geng – для неориентированных графов.

Языком программирования был выбран Kotlin. Кроссплатформенность языка позволяет запускать программу на любой ОС, на которую можно поставить JVM

Для удобной разработки используется система управления версиями Git – это бесплатная распределенная система управления версиями с открытым исходным кодом, предназначенная для быстрой и эффективной обработки всего, от небольших до очень крупных проектов.

В качестве протокола передачи данных был выбран gRPC – современная высокопроизводительная система удаленного вызова процедур (RPC) с открытым исходным кодом, которая может работать в любой среде. Он может эффективно соединять службы внутри и между центрами обработки данных с подключаемой поддержкой балансировки нагрузки, трассировки, проверки работоспособности и аутентификации. В gRPC клиентское приложение может напрямую вызывать метод серверного приложения на другом компьютере, как если бы это был локальный объект, что упрощает создание распределенных приложений и служб. gRPC дает множество преимуществ. В отличие от REST, он может максимально использовать HTTP 2, используя мультиплексированные потоки и следуя двоичному протоколу. Кроме того, он предлагает преимущества в производительности за счет структуры сообщений Protobuf, что очень важно в распределенных вычислениях, и удобстве использования, за счет встроенной функциях генерации кода, которая обеспечивает многоязычную среду.

2.3 Модули системы

Модуль **proto** содержит описание взаимодействия клиента и сервера – буферы протоколов – язык описания интерфейсов системы удаленного вызова процедур gRPC, работающего на протоколе HTTP/2. Эти преимущества позволяют оптимизировать затраты по времени на передачу данных в отличии от обычного HTTP API и json сериализации данных. Другие общие классы описаны в модуле **core**.

Модуль **graphtask** содержит интерфейс для алгоритма инварианта, который получает на вход граф в формате graph6, а на выход отдает вычисленное значение инварианта. При помощи имплементации этого интерфейса каждый желающий может написать свой алгоритм для подсчета своего инварианта на языке, компилируемом в java-байткод, и предоставить его на сервер, чтобы в последующем запустить его подсчет. Интерфейс нужен для подгрузки нового метода подсчета инварианта программой-клиентом прямо в процессе работы программы. Исходный код модуля приведен в приложении А.

Модуль **server** представляет собой консольное приложение, размещающееся на машине-сервере. Стандартный поток ввода используется для подачи списка графов, для которых нужно подсчитать их инварианты, а через параметры задается метод подсчета, который должен содержаться на сервере в виде скомпилированного файла, реализующего интерфейс из модуля **graphtask.**

Модуль **client** размещается на любом персональном компьютере, который хочет участвовать в вычислениях. Представляет собой консольное приложение, запускаемое с параметрами – адресом сервера и портом выполнения программы на сервере. Также есть возможность использовать версию, собранную в исполняемый файл exe, если машина-клиент не имеет установленной jvm.

2.4 Модуль client

Для абстрагирования вычислений инвариантов и выполнения задачи поиска графов с заданными свойствами были созданы два интерфейса, доступных для подключения другими пользователями и создания своих реализаций. Данные интерфейсы содержатся в отдельном репозитории системы.

2.5 Структура связи клиент-сервер. Загрузка новых инвариантов

Так как реализуем систему добровольных вычислений, то в связке клиент – сервер сервер должен играть центральную роль.

Задача загрузки на клиент новых алгоритмов для подсчета была решена с помощью возможности JVM подгружать классы в работающее приложение на лету. Таким образом возможно дополнить запущенную программу классом из отдельного исполняемого файла на лету, скрыв его реализацию под интерфейсами, описанными в пункте 2.4.

2.6 Client – сетевой слой

В данном разделе описывается реализация сетевого взаимодействия на стороне клиента.

2.7 Server – сетевой слой

В данном разделе описывается реализация сетевого взаимодействия на стороне сервера.

2.8 Дополнительный функционал системы

В данном разделе описывается возможность использования системы для решения задачи поиска графов, удовлетворяющих определенному условию, а также способ использования системы для собственных вычислений без использования сервера.

2.9 Работа системы

В данном разделе описываются способы запуска системы с различными параметрами, приводятся примеры запусков, а также результаты тестирования системы при вычислении различных инвариантов.

Результаты запуска системы для вычисления хроматического индекса приведены в таблице 1.

Таблица 1 – Результаты тестирования системы при подсчете хроматического индекса

Количество вершин	Сервер/кол-во частей	Клиенты	Время
9	L/1000	Vm, Pm	2 мин 42 сек
9	L/100	Vm, Pm	3 мин 21 сек
9	L/100	Vm	4 мин 4 сек
9	P/100	Pm	4 мин 25 сек
10	L/1000	Pm	1 ч 47 мин
11	L/1000	Pm	52 ч 37 мин

Результаты подтверждают, что хроматический индекс – достаточно сложный и неравномерный инвариант. Стоит отметить что под конец вычислений в многопоточном режиме процессор компьютера занят на 10-20%, что свидетельствует об активности всего лишь нескольких потоков, которые досчитывают свои группы графов, в то время как все остальные графы уже подсчитаны.

При подсчете индекса палитры интересно следующее наблюдение. При увеличении числа вершин у графа, увеличивается количество графов, для которых индекс палитры достигается на большем количестве цветов ребер. Такие графы, до 6 вершин включительно, приведены в таблице 2, а с количеством вершин 7 таких графов уже намного больше – 112.

Таблица 2 – Графы с количеством вершин до 6 включительно, у которых индекс палитры требует больше цветов ребер, чем минимальная раскраска

Кол-во вершин	Граф	Индекс палитры	Кол-во цветов в раскраске	Хроматический индекс
5	D]{	4	5	4
5	D~{	4	6	5
6	EErW	5	5	4
6	ETmw	5	6	5
6	ETnw	5	6	5

ЗАКЛЮЧЕНИЕ

Была рассмотрена задача разработки распределенной системы для вычисления инвариантов графов. Такая система была реализована на языке Kotlin. Реализованная распределенная система подсчетов инвариантов графов и запуска задач поиска графов позволяет проводить исследование теоретических проблем, формировать гипотезы и искать контрпримеры. Система является открытой, для примера ее использования был реализован модуль вычисления инвариантов хроматического индекса, индекса палитры, кликового числа, однако любой пользователь для расширения этой системы может реализовать задачу для вычисления определенного инварианта.

В ходе работы с помощью системы были подсчитаны значения индекса палитры для всех графов с количеством вершин от 1 до 7.

Систему может использовать любой пользователь, имеющий установленную JVM и утилиту geng из пакета программ Nauty на своем компьютере, для подключения к серверу в качестве клиента добровольных вычислений, либо для использования системы для собственных нужд.