Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

СРАВНИТЕЛЬНЫЙ АНАЛИЗ И ХАРАКТЕРИСТИКА АЛГОРИТМОВ О НАХОЖДЕНИИ ВЕРШИННОГО ПОКРЫТИЯ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Ищук Марии Алексеевны

Научный руководитель:
Зав.кафедрой, к.ф.-м.н. М.В. Огнева
(подпись, дата)

Зав. кафедрой:
К.ф.-м.н. М.В. Огнева
подпись, дата

Саратов 2017

ВВЕДЕНИЕ

Актуальность темы. Теория графов имеет большое теоретическое и практическое значение. Теоремы и алгоритмы теории графов используются не только в математике и компьютерных науках, но также в биологии, социологии, экономике и других областях. Некоторые задачи, которые исследуются в теории графов, имеют довольно трудные решения, которые, иногда, невозможно решить за полиномиальное время. Одной из таких является задача о нахождении вершинного покрытия.

Эта задача относится к классу NP-трудных. Несмотря на сложность таких задач, они очень часто встречаются в реальной жизни, ведь многое в реальном мире трудно объяснить с помощью элементарных алгоритмов. Например, кратчайшее решение «пятнашек», или задача коммивояжёра.

Таким образом, невозможно найти идеальное решение для задачи о вершинном покрытии, которое всегда выполнялось бы за минимально возможное время и давало наилучший результат. Однако существует несколько алгоритмов, которые дают приближенное решение. Все они имеют похожую идею, и различаются между собой в основном подходом к выбору вершин для добавления в вершинное покрытие. Эти алгоритмы и будут рассмотрены в данной работе.

Цель бакалаврской работы – изучение алгоритмов решения задачи о вершинном покрытии, их программная реализация и сравнительный анализ.

Поставленная цель определила **следующие задачи**:

1. Изучить литературу по данной теме;
2. Рассмотреть задачу о нахождении вершинного покрытия;
3. Разобрать и реализовать алгоритмы решения данной задачи;
4. Разработать параллельную версию алгоритмов, для которых это возможно;
5. Сравнить работу различных алгоритмов по нескольким параметрам.

Методологические основы разработки алгоритмов представлены в работах Разборова А. А. [3], Гэри М., Джонсона Д. [6], Ахо А., Хопкрофта Дж., Ульмана Дж. [8], Скиены С. [10].

Структура и объём работы. Бакалаврская работа состоит из введения, 5 разделов, заключения, списка использованных источников и 2 приложений. Общий объем работы – 67 страниц, из них 51 страница – основное содержание, включая 25 рисунков и 5 таблиц, список использованных источников информации – 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «ГРАФ. ОБЩИЕ СВЕДЕНИЯ» посвящен самим графам, предмету изучения теории графов.

В разделе описаны основные понятия и определения.

Граф представляет собой совокупность непустого множества вершин и наборов связей между ними [1].

Вершина покрывает ребро (или ребро покрывает вершину), если они инцидентны.

Также, в разделе описаны виды графов и способы их задания [2]:

1. Ориентированный граф;
2. Неориентированный граф;
3. Взвешенный граф;
4. Связный граф;
5. Полный граф;
6. Двудольный граф;
7. Эйлеров граф;
8. Гамильтонов граф.

Второй раздел «КЛАСС NP» посвящен описанию класса NP, а также задач, которые ему принадлежат.

NP (nonlinear polynomial – нелинейный полиномиальный) – это класс задач, решаемых за полиномиальное время. Стоит заметить, что время выполнения алгоритмов решения таких задач не обязательно будет расти от объема входных данных [3].

Примеры задач класса NP [4]:

1. Проблема выполнимости булевых формул (SAT);
2. Проблема существования гамильтонова цикла в графе;
3. Существование целочисленного решения системы линейных неравенств.

Существует класс задач, называемых NP-трудными. Это такие задачи, к которым сводимы все задачи из класса NP. То есть, существует полиномиальная функция, которая отображает экземпляр задачи класса NP к экземпляру NP-трудной задачи. Причем, решение обеих задач одинаково [5].

В программировании знание о том, что задача является NP-полной очень полезно. Так как при программной реализации любой задачи, сначала нужно найти её алгоритмическое решение, а также доказать или проверить, что оно является верным. В случае, если программисту будет неизвестно о принадлежности задачи к классу NP-полных, он может потратить очень много времени на поиски алгоритма, решаемого за полиноминальное время, а его и вовсе не существует.

Третий раздел «ЗАДАЧА О ВЕРШИННОМ ПОКРЫТИИ» посвящен постановке задаче о вершинном покрытии, а также алгоритмам ее решения.

Вершинное покрытие для неориентированного графа $G = (V, E)$ это множество его вершин S, такое что, у каждого ребра графа хотя бы один из концов входит в S [6].

«Ленивый» алгоритм для вершинного покрытия заключается в следующем [7]:

На каждом шаге, пока не кончатся все ребра в графе:

1. Выбираем случайное ребро графа $e = (u, v)$;
2. Добавляем в решение одну из выбранных вершин u или v;
3. Удаляем из графа все ребра, инцидентные вершинам u или v.

«Ленивый» алгоритм является 2–приближенным (превышает размер оптимального покрытия не более чем в два раза) алгоритмом с полиномиальным временем работы [8].

«Жадный» алгоритм [9]:

На каждом шаге, пока не кончатся все ребра в графе:

1. Выбираем вершину, покрывающую наибольшее число ребер;
2. Добавляем ее в решение S;
3. Удаляем ее из графа и все ребра, ей инцидентные.

Первая версия модификации «жадного» алгоритма [11]:

1. Вершинное покрытие пустое;
2. Пока количество вершин в нем меньше количества вершин в графе, и граф не пустой:
   2.1. Выбрать вершину второй степени в графе;
   2.2. Добавить ее в вершинное покрытие;
   2.3. Если смежные с ней вершины имеют степень больше второй, добавить ее в вершинное покрытие, иначе добавить любую смежную вершину;
3. Полученное вершинное покрытие – искомое.

Вторая версия модификации «жадного» алгоритма:

Пусть граф имеет n компонент связности, тогда:

1. Для всех вершин из компонент связности [1..n]

1.1. Вершинное покрытие пустое

1.2. Пока количество вершин в нем меньше количества вершин в графе, и граф не пустой:

1.2.1. Выбрать вершину второй степени в графе;

1.2.2. Добавить ее в вершинное покрытие;

1.2.3. Если смежные с ней вершины имеют степень больше второй, добавить ее в вершинное покрытие, иначе добавить любую смежную вершину;

1.2.4. Полученное вершинное покрытие – искомое, сохраняем его.

2. Объединяем все найденные вершинные покрытия.

Помимо этого, приведена сложность выполнения нескольких других существующих алгоритмов [11-15].

Третий раздел «ПРОБЛЕМА РАСПАРАЛЛЕЛИВАНИЯ ПРОГРАММ» посвящен описанию способов распараллеливания и общей теории на эту тему.

Распараллеливание программ — процесс адаптации алгоритмов, записанных в виде программ, для их эффективного исполнения на вычислительной системе параллельной архитектуры [16].

Существуют разные уровни распараллеливания [17-19]:

1. Распараллеливание на уровне задач;
2. Параллелизм данных;
3. Распараллеливание алгоритмов.

Так как для реализации алгоритмов для решения задачи о нахождении минимального вершинного покрытия был выбран язык C#, платформа .NET, описаны возможности распараллеливания в данной платформе. [20]

Четвертый раздел «ПРОГРАММНОЕ РЕШЕНИЕ ЗАДАЧИ О ВЕРШИННОМ ПОКРЫТИИ» посвящен реализации программного решения задачи о нахождении минимального вершинного покрытия.

Описан способ представления данных во входном файле, а также приведена часть кода, которая содержит основные переменные и способ представления данных в программном виде.

Перечислены виды графов, на которых производятся вычисления, с объяснением того, почему были выбраны именно они.

Реализованы два классических алгоритма, «ленивый» и «жадный», проведены эксперименты на различных видах графов, проанализировано время выполнения и размер полученного вершинного покрытия.

Помимо классических алгоритмов реализована параллельная версия «жадного» алгоритма и его модификация для графа, который содержит несколько компонент связности.

По итогам запусков различных алгоритмов для нескольких видов графов представлены сводные таблицы, на которых хорошо видно разницу во времени выполнения и эффективность в нахождении минимального вершинного покрытия.

ЗАКЛЮЧЕНИЕ

В ходе работы была изучена задача о нахождении минимального вершинного покрытия, а так же алгоритмы для её решения. Это «ленивый» и «жадный», а также модификация «жадного» алгоритма. Они были реализованы на языке C#. Все вариации «жадного» алгоритма также были реализованы параллельно. Были проведены несколько серий выполнения программы на графах различного вида, что влияло на время выполнения программы. После сравнения результатов работы всех алгоритмов, можно сделать вывод, что «жадный» алгоритм является лучшим для всех видов графов. Так как найденное вершинное покрытие действительно является меньше, чем находит «ленивый» алгоритм. К тому же, он затрачивает меньше времени на поиск минимального вершинного покрытия.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Уилсон Р. Введение в теорию графов. — М.: Мир, 1977.
2. Берж К. Теория графов и её приложения. — М.: ИЛ, 1962.
3. Разборов А. А. Алгебраическая сложность. — М.: МЦНМО, 2016.
4. Левин Л.А. Универсальные задачи перебора. — Проблемы передачи информации. — 1973.
5. Кононов А.В., Кононова П.А. Приближенные алгоритмы для NP-трудных задач. – Новосиб. гос. ун-т. — Новосибирск: РИЦ НГУ, 2014.
6. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982
7. Статья «Задача о вершинном покрытии» [Электронный ресурс]. URL: https://habrahabr.ru/post/120328/ (дата обращения: 28.11.2016).
8. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979.
9. Лекция «Независимые множества, клики, вершинные покрытия» [Электронный ресурс]. URL: http://www.intuit.ru/studies/courses/101/101/lecture/2959 (дата обращения: 28.11.2016).
10. Скиена С. Алгоритмы. Руководство по разработке. — СПб.: БХВ-Петербург, 2011. — 356 с.
11. Статья Vertex cover: further observations and further improvements, J. Chen, I. Kanj, and W. Jia, 2001.
12. Статья J. Buss and J. Goldsmith. Nondeterminism within P, 1993.
13. Статья L. Chandran and F. Grandoni. Refined memorisation for vertex cover, 2004.
14. Статья J. Chen, I. Kanj, and G. Xia. Improved Parameterized Upper Bounds for Vertex Cover, 2006.
15. В.А. Вальковский. Распараллеливание алгоритмов и программ. Структурный подход. – М.: Радио и связь.
16. Деннинг П. Дж., Браун Р. Л. Операционные системы // Современный компьютер. — М., 1986.

17. Шоу А. Логическое проектирование операционных систем. — М.: Мир, 1981.
18. Эндрю Таненбаум, Мартин ван Стеен. Распределенные системы. Принципы и парадигмы. — Санкт-Петербург: Питер, 2003.
19. Воеводин В. Параллельные вычисления. — СПб.: БХВ-Петербург, 2004.
20. Просиз Дж. Программирование для Microsoft .NET — М.: Русская редакция, 2003.