МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

Задача о назначениях: реализация и сравнительный анализ алгоритмов

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Зюзина Максима Сергеевича

Научный руководитель:
Зав. кафедрой ИиП
к.ф.-м.н., доцент _____ М.В. Огнева

подпись, дата

Зав. кафедрой
к.ф.-м.н. _____ М.В. Огнева

подпись, дата

Саратов 2019

ВВЕДЕНИЕ

Актуальность темы. Задача о назначениях является частным случаем классической транспортной задачи и, как следствие, является задачей транспортного типа.

Транспортная задача — задача о наиболее экономном плане перевозок однородного или взаимозаменяемого продукта из пункта производства (станций отправления), в пункты потребления (станции назначения) — является важнейшей частной задачей линейного программирования, имеющей обширные практические приложения не только к проблемам транспорта.

Схематично условие такой задачи может быть представлено в виде таблицы весов или стоимостей. Числа могут выражать, например, длину пути курьера до места назначения или среднюю оценку работника согласно мнениям его коллег.

В современных условиях развития каждое предприятие стремится с наименьшими затратами функционировать в сложившихся условиях с целью получения высоких доходов. Экономико-математические задачи о назначениях позволяют найти оптимальный вариант размещения одного кандидата на выполнение одной работы таким образом, чтобы минимизировать суммарные затраты по выполнению комплекса работ группой исполнителей. Также возможны некоторые модификации задачи о назначениях: во-первых, она иногда формулируется как задача максимизации (например, суммарного дохода от назначения всех исполнителей на работы); во-вторых, штатный состав организации может быть представлен большим количеством исполнителей, нежели количество работ, на которые должны быть назначены или, наоборот, большее количество работы, при недостаточном количестве исполнителей для ее выполнения; в-третьих, выполнение какой-либо работы по каким-либо причинам запрещается исполнять какому-либо работнику. В такой постановке данная задача

относится к классу комбинаторных, решение которых путем прямого перебора невозможно при достаточно больших n, так как число вариантов назначений составляет n!.

Цель бакалаврской работы — сравнить эффективность алгоритмов задачи о назначениях.

Поставленная цель определила **следующие задачи**:

1. Рассмотреть теоретические основы задач о назначениях;
2. Разобрать и реализовать алгоритмы:
    - венгерский алгоритм
    - алгоритм Хопкрофта-Карпа
    - метод ветвей и границ
    - алгоритм полного перебора
3. Протестировать и сравнить эффективность алгоритмов.

Методологические основы разработки и анализа алгоритмов представлены в работах Седжвика Р. [2], Скиены С. [3], Кормена Т. [4], Харари Ф. [5], Гэри М., Джонсона Д. [13], Клейнберга Дж., Тардос Е. [15].

Теоретическая и/или практическая значимость бакалаврской работы. Заключается в исследовании алгоритмов решения задачи о назначениях, возможность понять, какой из алгоритмов выгоднее использовать при решении реальных задач.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и семи приложений. Общий объем работы — 63 страницы, из них 41 страница — основное содержание, включая 12 рисунков и 8 таблиц, 20 страниц приложений, список использованных источников информации — 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Теоретические основы задач о назначениях» посвящен основным понятиям и определениям для решения задачи о назначениях, разбору основных алгоритмов решающих исследуемую задачу. Кроме того в этом разделе даны возможные формулировки задачи о назначениях.

Формулировка задачи:

Назначение на должность: имеются вакантные должности и претенденты на них, о каждом претенденте известно какие должности он может занять, требуется заполнить максимум вакансий (совместительство не допускается).

Более формально:

Дана матрица A размера $n \times n$, где n — это количество агентов(строки) и n — количество задач(столбцы), числа на пересечении строк и столбцов являются численной характеристикой(стоимость, длинной и т.д.). Требуется выбрать n чисел так, чтобы в каждой строке и каждом столбце находилось число, и при этом сумма выбранных чисел была минимальна.

Стоит отметить, что если количество не равно количеству заданий, иными словами матрица $n \times m$, то в таблицу можно добавить фиктивные строки или столбцы с фиктивными значениями, что в итоге, никак не повлияет на конечный результат. В общем случае, предполагается, что всегда $m \geq n$.

По своей сути, матрица представляет собой полный взвешенный граф и, таким образом, задача сводится к нахождению совершенного паросочетания в двудольном графе с количеством вершин $2n$.

Пусть V — непустое конечное множество вершин, V^2 — множество всех двухэлементных подмножеств из V. Обыкновенным графом G называется пара множеств (V, E), где E — произвольное подмножество из V^2. Если $E \subseteq V \times V$, то элементы множества E называются дугами, а сам граф G — ориентированным. Если же $E \subseteq \{\{x, y\}: x, y \in V \wedge x \neq y\}$, то

элементы множества E называют рёбрами, а граф G неориентированным[1].

Матрицей смежности A=||αi,j|| взвешенного графа $G = (V, E)$ называется матрица $A_{[V \times V]}$, в которой $\alpha_{i,j}$ — вес ребра, соединяющего вершины v_i и v_j.

Плотный граф — граф, в котором число рёбер E близко к максимально возможному у полного графа с числом V вершин: $E = \frac{V(V-1)}{2}$.

Путем в графе называется последовательность вершин, в которой каждая следующая вершина (после первой), является смежной с предыдущей вершиной на этом пути. Все вершины и ребра, составляющие простой путь, различны. Циклом называется простой путь, начальная и конечная вершины которого совпадают[2].

Граф $G = (V, E)$ называется двудольным, если множество его вершин V можно разбить на 2 непересекающихся множества V_1 и V_2 таких, что каждое ребро графа имеет одну вершину в V_1, а другую в V_2[3].

Паросочетанием в графе $G = (V, E)$ называется подмножество его рёбер такое, что каждой вершине инцидентно не более одного ребра. Паросочетание, содержащее наибольшее количество ребер, называется наибольшим, а такое паросочетание M в графе G, которое не содержится ни в каком другом паросочетании этого графа, то есть к нему невозможно добавить ни одно ребро, которое бы являлось несмежным ко всем рёбрам паросочетания называется максимальным.

Число рёбер в наибольшем паросочетании графа G называется числом паросочетания.

Вершины двудольного графа, инцидентные рёбрам паросочетания P, называются покрытыми, а неинцидентные — свободными.

Простая цепь в G называется чередующейся относительно P, если из любых двух соседних ребер этой цепи ровно одно лежит в P.

Чередующаяся цепь — путь в двудольном графе, для любых двух соседних рёбер которого верно, что одно из них принадлежит паросочетанию P, а другое нет.

Уменьшающая цепь — чередующаяся цепь, у которой оба конца покрыты.

Сбалансированная цепь — чередующаяся цепь, у которой один конец свободен, а другой покрыт.

Числом рёберного покрытия называется размер минимального рёберного покрытии графа G и обозначается через ρ(G).

Сетью называется ориентированный граф $G = (V, E)$, каждому ребру $(u, v) \in E$ которого, поставлено в соответствие вещественное число $c(u, v) \geq 0$, называемое пропускной способностью ребра. В случае, если $(u, v) \notin E$, то $c(u, v) = 0$. В графе выделены две вершины: исток s и сток t.

Вершинным покрытием графа $G = (V, E)$ называется такое подмножество S множества вершин графа V, что любое ребро этого графа инцидентно хотя бы одной вершине из множества S.

Минимальным вершинным покрытием графа $G = (V, E)$ называется вершинное покрытие, состоящее из наименьшего числа вершин.

Алгоритм полного перебора

Во многих задачах из различных областей знания ставятся вопросы-задания типа: «Сколько существует способов …», «Подсчитайте количество элементов …», «Перечислите все возможные варианты …», «Есть ли способ …», «Существует ли объект…» и т. п. Ответы на них, как правило, требуют исчерпывающего поиска в некотором множестве M всех возможных вариантов, среди которых находятся решения конкретной задачи. Существуют два общих метода организации исчерпывающего поиска: перебор с возвратом и его естественное логическое дополнение — метод решета.

Решение задачи методом перебора с возвратом строится конструктивно

последовательным расширением частичного решения.

Фрагмент кода алгоритма перебора может быть представлен таким образом:

```
        private static void _solution(int n, int m,
List<List<int>> a, int k)
        {
// перебор по всем агентам
            for (int i = k; i < n; i++)
            {
// перебор по всем задачам
                for (int j = 0; j < m; j++)
                {
// назначение незанятой задачи агенту
                    if (!usedTask[j])
                    {
                        cur_sum += a[i][j];
                        usedTask[j] = true;

                        if (k + 1 < n)
                            _solution(n, m, a, k + 1);

                        // возвращаем прежние значения "после
отката" рекурсии
                        usedTask[j] = false;
                        cur_sum -= a[i][j];

                        //прекращаем поиск незанятой задачи для
работника
                        break;
                    }
                }
```

```
                if (i == n - 1)
// текущий ответ сравнивается с предполагаемым результатом
                    result = Min(result, cur_sum);
            }
        }
```

Венгерский алгоритм

Описание алгоритма

- **Шаг 0.** Введем следующее понятие:

  Назовём *потенциалом* два произвольных массива чисел $u[1 \dots n]$ и $v[1 \dots n]$ таких, что выполняется условие:

  $u[i] + v[j] \leq a[i][j],\ i = \overline{1,n}$ где a — заданная матрица.

  - **Шаг 1.** Добавляем в рассмотрение очередную строку матрицы a.
  - **Шаг 2.** Пока нет увеличивающей цепи, начинающейся в этой строке, пересчитываем потенциал.
  - **Шаг 3.** Как только появляется увеличивающая цепь, чередуем паросочетание вдоль неё (включая тем самым последнюю строку в паросочетание), и переходим к началу (к рассмотрению следующей строки).

  Конец

  Метод ветвей и границ

  Метод состоит из двух этапов. Первый этап:

1. Вычисляется наименьший элемент в каждой строке матрицы (константа приведения для строки).
2. Осуществляется переход к новой матрице затрат: из каждой строки матрицы вычитается ее константа приведения.
3. Вычисляется наименьший элемент в каждом столбце (константа приведения для столбца).
4. Осуществляется переход к новой матрице затрат: константа приведения вычитается из каждого столбца матрицы. Как результат

получается матрица затрат, в которой в каждой строчке и в каждом столбце имеется хотя бы один нулевой элемент.

5. На данном этапе вычисляется граница, как сумма констант приведения для столбцов и строк (данная граница будет являться стоимостью, меньше которой невозможно построить искомый маршрут)[16].

Второй этап:

1. Вычисляется штраф за неиспользование для каждого нулевого элемента приведенной матрицы затрат. Штраф за неиспользование элемента с индексом (h, k) в матрице, означает, что это ребро не включается в наш маршрут, а значит минимальная стоимость «неиспользования» этого ребра равна сумме минимальных элементов в строке h и столбце k.

a) Ищем все нулевые элементы в приведенной матрице.

b) Для каждого из них считаем его штраф за неиспользование.

c) Выбираем элементы, которым соответствует максимальный штраф

2. Все множество маршрутов разбивается на два множества: содержащие

ребра с максимальным штрафом и не содержащие эти ребра.

3. Вычисляются оценки затрат для маршрутов, входящих в каждое из этих множеств:

a) Для множеств, не содержащих ребра с максимальным штрафом: ребро (h_i, k_i) с максимальным штрафом не берется, для него оценка затрат равна оценки затрат множества всех маршрутов, плюс штраф за неиспользование ребра (h_i, k_i).

b) При вычислении затрат для множеств содержащих ребра с максимальным штрафом: ребро (h_i, k_i) входит в маршрут, значит ребро (k_i, h_i) в маршрут входить не может, поэтому в матрице затрат присваивается c(k_i,

h_i) = infinity, а так как пункт h_i и k_i уже пройдены, то ни одно ребро, выходящее из h_i, и ни одно ребро, приходящее в k_i, уже использоваться не могут. Поэтому из матрицы затрат вычеркивается строка h_iи столбец k_i. После этого матрица приводится, и тогда оценка затрат для множеств содержащих ребра с максимальным штрафом равна сумме оценки затрат для множества всех маршрутов и r(h_i, k_i), где r(h_i, k_i)— сумма констант приведения для измененной матрицы затрат[17].

4. Из всех неразбитых множеств выбирается то, которое имеет наименьшую оценку. Так продолжаем, пока в матрице затрат не останется одна не вычеркнутая строка и один не вычеркнутый столбец.

Алгоритм Хопкрофта-Карпа

Алгоритм работает следующим образом:

0. вводится фиктивная вершина v_{-1};

1. с помощью поиска в ширину, множество вершин V_1 первой доли графа разбивается на слои. Изначально все вершины $v \in V_1$ являются свободными, т.е. ни одно инцидентное им ребро не принадлежит паросочетанию, и образуют единый монолитный слой. В свою очередь, введенная фиктивная вершина принадлежит слою уровнем выше по отношению к первой найденной свободной вершине. На последующих итерациях, начиная со свободных вершин, поиск в ширину пересчитывает номера слоев для вершин, смежных с ними и далее смежных всем рассматриваемым вершинам, в соответствии со следующей формулой: $d[v] = d[u] + 1$, где $d[v_i]$ — номер слоя, в который помещена вершина v_i, $u \in V_1$ — рассматриваемая, изначально свободная вершина, $v \in V_1$ — вершина, смежная с вершиной $w \in V_2$ в случае, если ребро (u, w) принадлежит паросочетанию. Если же ребро (u, w) не принадлежит паросочетанию, то вершина u соответствует введенной фиктивной[20] вершине v_{-1}. Именно на этом этапе разбиение на слои будет завершено;

2. алгоритм производит увеличения паросочетания по рёбрам со свободными вершинами. Ниже приведен псевдокод алгоритма Хопкрофта-Карпа.

Второй раздел «Сравнительная характеристике алгоритмов задачи о назначениях» посвящен анализу времени работы реализованных алгоритмов.

Все алгоритмы прошли тестирование с замерами времени работы. Для получения практического времени работы алгоритмов использовался ноутбук со следующими характеристиками:

- процессор AMD A10-4600M 2.30 GHz
- оперативная память (Мб) — 8192.

Все измерения проводились с помощью класса System.Diagnostics.Stopwatch фреймворка .NET. Класс Stopwatch основан на HPET (High Precision Event Timer, таймер событий высокой точности)[21].

Для алгоритма полного перебора использовались размеры матриц — 5x5, 6x6, 7x7, 8x8, 9x9, 10x10, 11x11, 12x12, 13x13, виду временной сложности этого алгоритма. С увеличением размера задачи, время выполнения программы росло со скоростью роста функции факториала.

Венгерский алгоритм, метод ветвей и границ, алгоритм Хопкрофта-Карпа показывали пренебрежимо малое время работы на тех же размерах, поэтому были выбраны 10x10, 50x50, 75x75, 100x100, 200x200, 300x300, 400x400, 500x500, 800x800, 1000x1000.

Для генерации матриц произвольного размера использовались статические классы *GenMatrix0* и *GenMatrix1* — GenMatrix0 генерирует матрицу $n \times m$; GenMatrix1 генерирует матрицы размером $n+1 \times m+1$ с нулевой строкой и нулевым столбцом для удобства обработки алгоритмами.

Для замеров результатов работы каждого алгоритма, в главном классе программы *Program* реализованы соответствующие методы, которые

генерируют матрицы заданного размера, запускают алгоритм, выполняют подсчёт времени выполнения.

Все размеры содержатся в массиве *dimensions*. При необходимости тестирования матриц других размеров, достаточно их добавить в этот массив.

Тестирование

Тестирование производилось с помощью класса *CheckResults*, который сравнивал данные полученные от алгоритма с готовыми данными, которые предпосчитывались вручную для малых матриц, либо брались за основу уже готовые матрицы с заранее известным ответом[21][22].

ЗАКЛЮЧЕНИЕ

Задача о назначениях имеет множество приложений в реальной жизни, транспортных задачах, на таксомоторных предприятиях, при создании программа машинного зрения и в прочих подобных ситуациях. Отсутствие оптимального метода решения этой задачи и попытки его создать привели к появлению множества методов, которые заметно отличаются и по времени выполнения. На практике важно находить ответ за приемлемое время, потому что бизнес-процессы в современном мире накладывают жёсткие ограничения по времени на решение задачи. Для промышленных предприятий необходимо быстро производить расчёт, так как время простоя в ожидании результата может оказаться потерями больших сумм прибыли.

Каждый алгоритм был протестирован на квадратных матрицах размеров 10x10, 50x50, 75x75, 100x100, 200x200, 300x300, 400x400, 500x500, 800x800, 1000x1000. Для алгоритма полного перебора использовался набор — 5x5, 6x6, 7x7, 8x8, 9x9, 10x10, 11x11, 12x12, 13x13. Каждая матрица генерировалась с произвольными положительными числами.

Итог — для реальных промышленных условий с жёсткими рамками по времени подойдут венгерский алгоритм и алгоритм Хопкрофта-Карпа. В то

же время, реализация алгоритма Хопкрофта-Карпа отличается нетривиальным подходом и, вожможно, использовать метод ветвей и границ, который и является более медленным, но простым в реализации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. *Асанов М. О., Баранский В. А., Расин В. В.* Дискретная математика: графы, матроиды, алгоритмы. — Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001. — 288 с. Агальцов, В.П. Математические методы в программировании: учебник. В.П. Агальцов, И.В. Волдайская. - М.: ИД «ФОРУМ»: ИНФРА-М, 2006 г. - 224 с.: ил.

2. *Седжвик Р.* Фундаментальные алгоритмы на C++. Алгоритмы на графах: Пер. с англ. — СПб: ООО «ДиаСофтЮП», 2002. — 496 с.

3. *Скиена С.* Алгоритмы. Руководство по разработке. — 2-е изд.: Пер. с англ. — СПб.: БХВ-Петербург, 2011. — 720 с.

4. *Кормен Т. и др.* Алгоритмы: построение и анализ, 2-е издание: Пер. с англ. — М.: Издательский дом «Вильямс», 2005. — 1296 с

5. *Харари Ф.* Теория графов, 4-е издание. М.: Либроком — 302 с

6. *Кирсанов М. Н.* Графы в Maple. Задачи, алгоритмы, программы. — М.: Издательства ФИЗМАТЛИТ, 2007. — 168 с.

7. *Ахо А. и др.* Структуры данных и алгоритмы. : Пер с англ.: М. : Издательский дом «Вильямс», 2003 с. — 384 с.

8. *Кристофидес Н.* Теория графов. Алгоритмический подход. : Пер. с англ. : М.: Мир, 1978. — 432 с.

9. *Асанов М., Баранский В., Расин В.* — Дискретная математика: Графы, матроиды, алгоритмы — 2010, 368 стр.

10. Метод ветвей и границ. Задача о назначениях. Задача о рюкзаке. [Электронный ресурс]. URL: http://www.studfiles.ru/preview/3654868/ (Дата обращения: 23.09.2018)

11. Статья «Матрица смежности». [Электронный ресурс]. URL: http://ru.wikipedia.org/?oldid=68515092 (дата обращения: 26.07.2018).

12. *Актанорович С. В. и др.* Алгоритмы и структуры данных. Потоковые алгоритмы : метод. пособие по курсу «Теория графов. Потоковые алгоритмы» для студ. спец. I-31 03 04 «Информатика» всех форм обуч. — Минск: БГУИР, 2011. — 47 с.

13. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи: Пер. с англ. — М. : Мир, 1982. — 416 с.

14. *Бирюков А.С., Ищук М.А., Огнева М.В.* Использование алгоритмов теории графов в задачах по программированию повышенной сложности//Информационные технологии в образовании. 2016. с. 25-30.

15. *Клейнберг Дж., Тардос Е.* Алгоритмы: разработка и применение. Классика Computers Science / Пер. с англ. Е. Матвеева. — Спб.: Питер, 2016. — 800 с.

16. *Таха Хэмди А*. Введение в исследование операций. 6-е издание.: Пер. с англ.: — М.: Издательский дом «Вильямс», 2001. — 912 с.

17. *Банди Б.* Основы линейного программирования /Пер. с англ. - М.: Радио и связь, 1989. - 176с.

18. *Муравьев, В.И.* Модели и методы оптимизации в экономике и менеджменте: учеб. пос. для практ. занятий / В.И. Муравьев [и др.]; Балт. гос. техн. ун-т.– СПб., 2008.: 197 с.

19. *Липский В.* Комбинаторика для программистов: Перевод с польского — М.: «Мир», 1988. — 200 с.

20. *Зандер Е. В.* Исследование операций в экономике : учеб. пособие. — Сибирский федеральный ун-т. Красноярск: 2007. — 202 с.

21. Статья «Под капотом у Stopwatch». [Электронный ресурс]. URL: https://habr.com/ru/post/226279/ (дата обращения: 18.09.2018).

22. Math 20 - Introduction to Linear Algebra and Multivariable Calculus. [Электронный ресурс] URL:http://www.math.harvard.edu/archive

/20_spring_05/handouts/assignment_overheads.pdf (дата обращения: 23.11.2018).