UNIVERSITY NAME (IN BLOCK CAPITALS)

Исследование алгоритмов

by

Козловский Никита

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty Name
Department or School Name

Оглавление

1 Введение **1**

2 Обзор задачи **3**
2.1 Перестановки 4
2.1.1 Линейная задача о назначениях в терминах перестановок 5
2.1.2 Трехиндексная задача о назначениях 5

3 Алгоритм решения **7**
3.1 Обзор алгоритмов 7
3.2 Введение к алгоритму 8
3.3 Алгоритм 9
3.4 Блок-схема 9
3.5 Комментарии к алгоритму 11
3.6 Вводимые модификации 11
3.6.1 Генерация нескольких начальных перестановок 11
3.6.2 Выбор лучшей перестановки 11
3.7 Итеративный алгоритм 11

4 Программная реализация и вычислительный эксперемент **12**
4.1 Описание 12
4.2 Эксперимент 12

Глава 1

Введение

Одной из фундаментальных задач комбинаторной оптимизации является задача о назначениях (ЗОН). В своей классической постановке эта задача звучит так:

Имеется некоторое число работ и некоторое число исполнителей. Любой исполнитель может быть назначен на выполнение любой (но только одной) работы, но с неодинаковыми затратами. Нужно распределить работы так, чтобы выполнить работы с минимальными затратами.

Так как в данной форме рассматривается 2 множества – работников X и работ Y, затраты могут быть выражены ввиде $(c_ij) \in$ A, где A матрица из $Matr_{n\times n}$ и такая задача называется двухиндекской.

В 1955 Куном был опубликовано решение этой задачи [link] в виде Венгерского алгоритма. В 1957 Манкрес определил, что алгорим является строго полиномианльным, а Карп улучшил его, добившись временной сложности $O(n^3)$

Естественно обобщить эту задачу, рассмотрев многоиндексную задачу о назначениях. Однако, уже для трехиндексной ЗОН было показано [кем?], что она принадлежит к классу нп-полных, т.е. не может быть решена за полиномиальное время.

Соответсвенно возникает проблема выбора достаточно хорошего решения. Само собой, эта задача, как и любая задача дискретной оптимизации, может быть решена полным перебором. Однако, слишком большая (экспоненциальная?) временная сложность для такого метода не позволяет использовать его в реальной жизни. Однако, имеет место улучшенная версия этого алгоритма – метод ветвей и границ. В худшем случае он сводится к полному перебору, но чаще требует гораздо меньшего числа операций [для получения *приближенного* решения – а не точный ли он?].

Большую практическую ценность представляют т.н. эвристические алгоритмы. Они за приемлимое время позволяют получить приближенное решение. Цель данной работы состоит в изучении одного из таких методов, для корого Гимади в [] было

показано, что решения, полученные с помощью такого алгоритма сходятся при $n \to$ inf. Для достижения этих целей необходимо решить следующие задачи:

- Изучение математической модели 3-АЗОН
- Изучить метод, предложенный Гимади
- Программно реализовать этот метод
- И провести его анализ

Глава 2

Обзор задачи

Задача о назначениях в самом общем виде формулируется следующим образом: есть некоторое количество *работ* и некоторе количество *исполнителей*. Любой исполнитель может быть назначен на выполнение любой (но только одной) работы, но с неодинаковыми затратами. Нужно распределить работы так, чтобы выполнить работы с минимальными затратами.

Строгая математическая формулировка звучит так.

Пусть даны множества A, T и функционал стоимости $C : A \times T \to \mathbb{R}$. Необходимо найти биекцию $f : A \to T$, такую что $\sum_{a \in A} C(a, f(a))$ минимальна.

Рассмотрим матрицу $C \in Matr_{n \times m}$, тогда $c_i j$ – стоимость назначения i работника на j работу, соотвественно целевая фунция переписывается ввиде $\sum_{a \in A} C_{a,f(a)}$.

Если число исполнителей и работ совпадает, $n = m$, то задачу называют линейной.

Эту задачу также можно переписать в виде задачи линейного программирования.

$$\sum_{i \in A} \sum_{j \in T} C(i,j) x_{ij}$$

и ограничениями

$$\sum_{j \in T} x_{ij} = 1 \text{ для } i \in A$$
$$\sum_{i \in A} x_{ij} = 1 \text{ для } j \in T$$
$$x_{ij} \geq 0 \text{ для } i, j \in A, T$$

В терминах общей формулировки первое ограничение означает, что на каждый работник может быть назначен лишь на одну работу, а второе означает, что каждая работа может быть отдана лишь одному работнику.

2.1 Перестановки

Введем понятие назначения. Мы можем представлять назначение как некое биективное отображение ϕ, которое ставит элементы конечного множества U в соотвествие элементам конечного множества V. В тоже время назначение является перестановкой, которая записывается в виде

$$\begin{pmatrix} 1 & 2 & \dots & \text{n} \\ \varphi(1) & \varphi(2) & \dots & \varphi(n) \end{pmatrix}$$

Каждой перестановке множества $\{1, 2, \dots, n\}$ соответсвует единственная матрица перестановок $\text{X}_\varphi \in \text{Matrix}_{n \times n}$, элементы котороый определяются как

$$x_{ij} = \begin{cases} 1 & \text{если } j = \varphi(i) \\ 0 & \text{иначе} \end{cases}$$

Пример

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 2 & 3 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

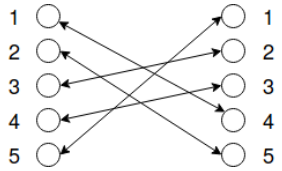

Рис. 2.1: A boat.

Обозначим множество S_n как множество всех возможных перестановок множества $\{1, 2, \dots, n\}$. Мощность этого множества $n!$.

2.1.1 Линейная задача о назначениях в терминах перестановок

Пусть дана матрица $n \times n$ весовых коэфициентов $C = (c_{ij})$. Требуется минимизировать линейную форму

$$\sum_{i=1}^{n} c_{i\varphi(i)}$$

то есть линейная задача о назначениях может быть поставлена в виде

$$\min_{\varphi \in S_n} \sum_{i=1}^{n} c_{i\varphi(i)}$$

.

При этом, если перестановки задаются матрицей перестановок X $= (x_{ij})$, линейная задача о назначениях может быть записана как задача линейной оптимизации

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.1}$$

$$\text{ограничения:} \sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, \ldots, n) \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, \ldots, n) \tag{2.3}$$

$$x_{ij} \in \{0, 1\} \qquad (i, j = 1, \ldots, n) \tag{2.4}$$

Ограничения (2) – (4) задают допустимое множество

В дальнейшем будем называть X матрицей назначений.

2.1.2 Трехиндексная задача о назначениях

Аксиальная трехиндесная задача о назначениях может быть определена следующим образом. Пусть даны n^3 весовых коэфициентов $c_{ijk}, (i, j, k = 1 \ldots n)$. Необходимо найти такие перестановки φ и ξ, что

$$\min_{\varphi, \xi \in S_n} \sum_{i=1}^{n} c_{i\varphi(i)\xi(i)}$$

где S_n множество всех перестановок целых чисел от $1 \ldots n$.

Так же задача может быть переписана как задача целочисленного линейного программирования.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}x_{ijk}$$

ограничения

$$\sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1 \qquad (j = 1, \ldots, n)$$

$$\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1 \qquad (i = 1, \ldots, n)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} = 1 \qquad (k = 1, \ldots, n)$$

$$x_{ijk} \in \{0, 1\} \qquad (i, j, k = 1, \ldots, n)$$

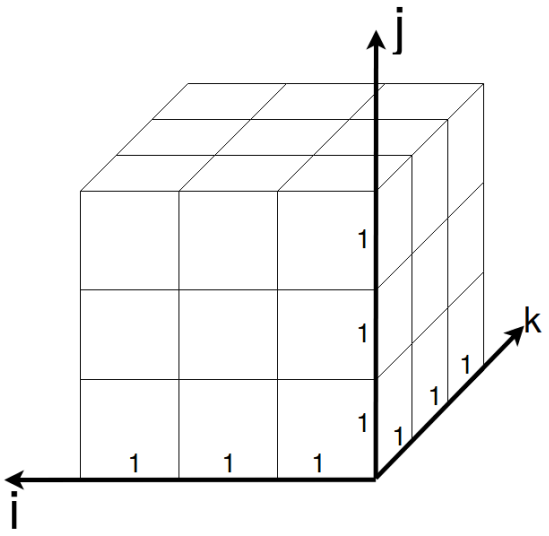

Рис. 2.2: A boat.

3-АЗОН состоит в том, чтобы выбрать среди элементов трехмерной матрицы $C = c_{ijk}$ такие n^2 элементов, что сумма в каждом выбраном сечении (при фиксированных i, j, k) минимальной.

Известна следующая интерпритация этой задачи. Необходимо назначить работника i на работу j на машине k так, чтобы стоимость назначения была минимальна.

Трехиндесная задача о назначениях также известна в планарной постановке. Содержательно она отличается от аксиальной тем, что что сумма всех выбранных элементов должна быть минимальна.

Глава 3

Алгоритм решения

3.1 Обзор алгоритмов

Задача о назначениях имеет $(n!)^2$ возможных решений. Трехиндексная задача, в отличие от линейной, не может быть решена за линейное время и принадлежит к классу нп полных. Это было показано Karp[109 1] (book1).

Эйлер [75 1] (book1) изучал политроп (выпуклая оболочка всех возможных решений задачи (2)) трехиндексной аксиальной задачи о назначениях. Были открыты неравенства, разделяющие классы граней (class of facets), с помощью которых можно было получить оценки точного решения задачи (неточно). Независимо Balas и Saltzman [16 1] предложили (n^4) алгоритм, который был улучшен Balas и Qi [15 1] до сложности (n^3)

Эвристический алгоритм решение 3-АЗН, т.е. включающий практический метод, не являющийся гарантированно точным или оптимальным, но достаточный для решения поставленной задачи, был первым предложен Pierskalla [142 1]. Классическим точным алгоритмом решения является метод ветвей и границ. Большая часть алгоритмов делят задачу на 2 подзадачи, в каждой из которой одна переменная x_{ijk} зафиксирована и равна 0 и 1 соотвественно, таким образом размерность подзадач уменьшается. Balas и Saltzman [17 1] разработали алгоритм, используя структуру задачи, может фиксисировать несколько переменных на одной ветви алгоритма. Burkard и Rudolf [44 1] поставили эксперементы с различными схемами решения 3-АЗН

Hansen и Kaufman [100 1] описали метод одновременного решения прямой и двойственной задачи, который похож на Венгерский алгоритм для линейной задачи о назначениях. Идея метода заключалась в использовании гиперграфа вместо двудольного графа.

Другой подход, удобный для получения нижних оценок 3-АЗН – представление (? функция, упорощение) Лагранжа. Внеся ограничения 3-АЗН в целевую функцию в виде множителей Лагранжа, можем получить представление задачи [*]

$$L(\psi,\xi) = min\{\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(c_{ijk}+\psi_j+\xi_i)x_{ijk} - \sum_{j=1}^{n}\psi_j\sum_{i=1}^{n}\xi_j\}$$

где

$$\sum_{i=1}^{n}\sum_{j=1}^{n}x_{ijk} = 1, k = 1,2,\ldots,n$$
$$x_{ikj} \in 0,1, \quad 1 \le i,j,k \le n$$
$$\psi \in \mathbb{R}^n, \xi \in \mathbb{R}^n$$

Так как $L(\psi,\xi)$ выпуклая функция, ее максимум может быть вычислен методом субградиентов. Frieze and Yadegar [83 1] описали вычислительный эксперемент, подтверждающие возможность применения подобного метода. Balas and Saltzman [17] улучшили результаты, рассмотрев другой вид функции Лагранжа.

В некоторых частных случаях возможно решение 3-АЗН. Например, Burkard, Rudolf, and Woeginger [45] рассматривали задачи с разделяемыми весовыми коэфициентами, $c_{ijk} = u_i v_j w_k$, где u_i, v_j, w_k – некоторые неотрицательные числа. В своей работе они показали, что задача на максимум решается за полиномиальное время, тогда как задача на минимум остается нп полной. Также за полиномиальное время решается задача на максимум, весовые коэфициенты которой берутся из массива Mogne [42] С друго стороны, Crama and Spieksma [61], рассматривая частные случаи 3-АЗН в терминах теории графов, обнаружили несколько вариантов 3-АЗН на минимум, решение для которых может быть получено за полиномиальное время.

3.2 Введение к алгоритму

Обозначим через f_{A} и $f*$ приближенное (полученное при помощи алгоритма А) и оптимальное значение целевой функции функции в некоторой конкретной задаче соответсвенно.

Будем говорить, что алгоритм А имеет оценки $(\epsilon_{\mathrm{A}}, \delta_{\mathrm{A}})$ если выполнено неравентво

$$Pr\{f_{\mathrm{A}} > (1+\epsilon_{\mathrm{A}} f*)\} \le \delta_{\mathrm{A}}$$

, где ϵ_{A} есть оценка относительной погрешности решния, получаемого алгоритмом А, δ_{A} – вероятность несрабатывания алгоритма А, что можно трактовать как долю случаев, когда алгоритм А не гарантирует точность в пределах ϵ_{A}.

Алгоритм A назвается асимптотически оптимальным если существую оценки $(\epsilon_{\mathrm{A}}, \delta_{\mathrm{A}})$ стремящиеся к нулю с ростом размерности.

В дальнейшем будем полагать, что элементы матрицы весовых коэфициентов c_{ijk} являются независимыми случайными величинами, которые выбираются из $[a_n, b_n]$, где $a_n > 0$ и функции распределения одинаковы и определяются как

$F_\xi(x) = Pr\xi < x$, где $\xi = (c_{ijk} - a_n)/(b_n - a_n)$ – нормализированная случайная переменная.

Через M_n обозначим множество всех матриц, определенных выше.

3.3 Алгоритм

Пусть $\phi = X \to \mathbb{N}$ – любая целочислено значащая функция, при этом $1 < \phi_n < n$

1. Берем произвольную подстановку $\pi \in S_n$. Пусть (d_{jk}) - $n \times n$ матрица, содержащая элементы исходной матрицы (c_{ijk}), где индекс $j = \pi(i)$ такой, что

   $$d_{ij} = c_{\pi^{-1}(j)jk}$$

   для любых $1 \leq j$,$n \leq n$ Положим $f = 0; j = 1; \mathrm{K} = 1, 2, \ldots, \phi_n$.

2. Выберем номер $\sigma(j)$ минимального элемента из множества $\operatorname{argmin} d_{jk} | k \in K$.

3. Полагаем $f = f + d_{j\sigma(j)}; \mathrm{K} = \mathrm{K} \setminus \sigma(j); k = j + \phi_n$

4. Если $k \leq n$, то $K = K \bigcap k$.

5. $j = j + 1$

6. Повторяем п.2, пока j<n. В противном случае идем к п.7

7. Результатом работы алгоритма $\mathrm{A}(\phi_n)$ является значение функции f целевой функции $f_{\mathrm{A}(\phi_n)}$.

3.4 Блок-схема

пока уезжает в конец текста

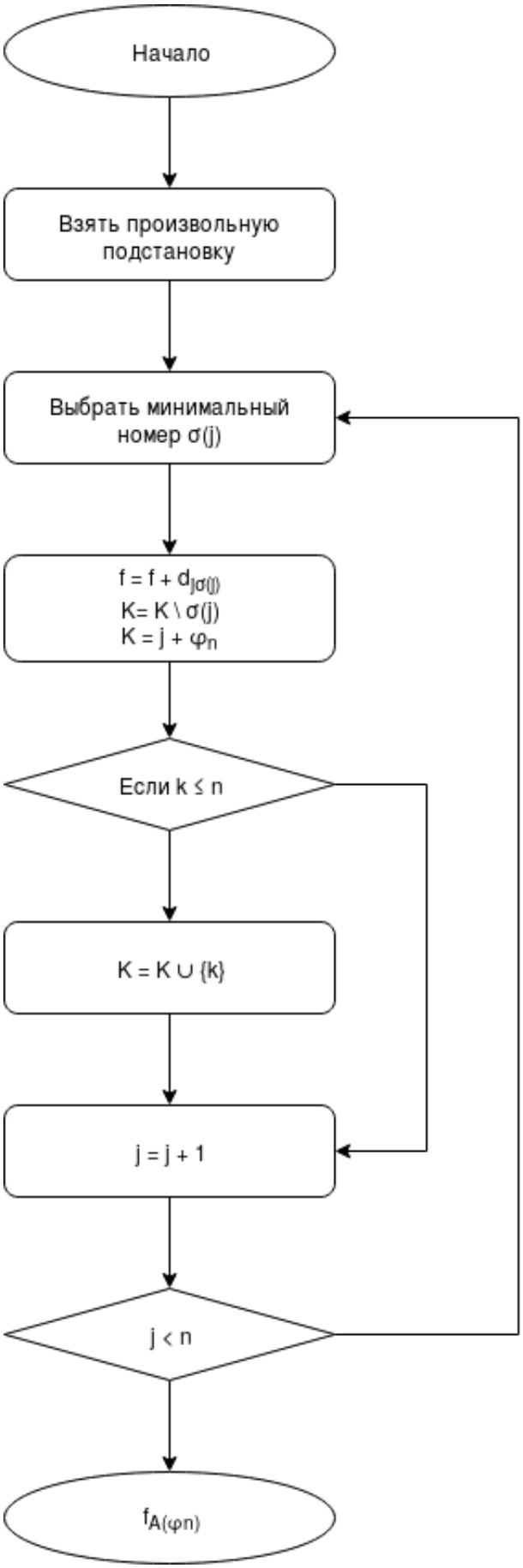
Начало
Взять произвольную подстановку
Выбрать минимальный номер σ(j)
f = f + d_jσ(j)
K= K \ σ(j)
K = j + φn
Если k ≤ n
K = K ∪ {k}
j = j + 1
j < n
fA(φn)

Рис. 3.1: A boat.

3.5 Комментарии к алгоритму

Асимптотическое поведение трехиндексной аксиальной задачи о назначениях значительно отличается от поведения классической задачи о назначениях. В ряде статей Grundel Krohmal Oleveira было изучено поведение ожидаемого значения оптимальной функции. Ими было доказано, что оно стремится к левой границе распределения весовых коэфициентов.

Теорема При $b_n/a_n = o(n/\ln n)$ алгоритм является ассимптотически оптимальным для 3-АЗН на классе матриц $_n$ и его временная сложность $O(n^2)$.

При $b_n/a_n = o(\ln n)$ алгоритм является ассимптотически оптимальным для 3-АЗН на классе матриц $_n$ и его временная сложность $O(n \ln n)$.

Несмотря на то, что алгоритм показывает полиномиальное время выполения, возможен ряд улучшений Например, заметно что алгоритм неустойчив относительно выбора начальной перестановки. Также,

3.6 Вводимые модификации

3.6.1 Генерация нескольких начальных перестановок

Изменим алгортим следующим образом

Пусть на начальном этапе дается не одна случайная перестановка, а m. Тогда на выходе можем выбрать лучшую ... Бред

3.6.2 Выбор лучшей перестановки

Введем функционал вида

При известном точном решении будем говорить, что одна перестановка лучше другой если она за тоже число шагов будет ближе сходится к точному решению

3.7 Итеративный алгоритм

Добавим следующие шаги После последнего шага сохраним результат, пойдем в начало и запустим алгоритм еще раз Повторим M раз Получим M выводов, в качестве ответа выберем устредненное значение.

Глава 4

Программная реализация и вычислительный эксперемент

4.1 Описание

здесь будет алгоритм, вводные данные

4.2 Эксперимент

здесь будут таблицы