Исследование и модификация алгоритма решения трехиндексной аксиальной задачи о назначениях с использованием случайных перестановок

выполнил Н.С. Козловский.

научный руководитель
к.ф.-м.н., С. Н. Медведев.

ВГУ, факультет ПММ
кафедра ВМиПИТ

июль, 2018

Цель работы

- Исследовать
- Модифицировать

эвристический (приближенный) алгоритм решения 3-ЗОН, основанного на сведении задачи к двухиндексной с использованием перестановок.

Задачи

- Изучить математическую модель 3-АЗОН
- Изучить и проанализировать метод метод, сводящий задачу к двухиндексной
- Разработать модификации данного алгоритма
- Программно реализовать данный алгоритм и провести вычислительный эксперимент

Постановка двухиндексной ЗОН

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.

Мат модель 2-ЗОН

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

$$\sum_{a \in A} C(a, f(a)) \tag{1}$$

минимальна.

Постановка в виде ЗЛП

Пусть $|\mathrm{A}| = |\mathrm{T}| = n$, C – матрица $n \times n$. Тогда ЗОН можнопредставить в виде задачи линейного програмирования

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2}$$

и ограничениями

$$\sum_{i=1}^{n} x_{ij} = 1 \tag{3}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \tag{4}$$

$$x_{ij} \in 0, 1 \text{ для } i = 1 \dots n, j = 1 \dots n \tag{5}$$

Алгоритмы решения ЗОН

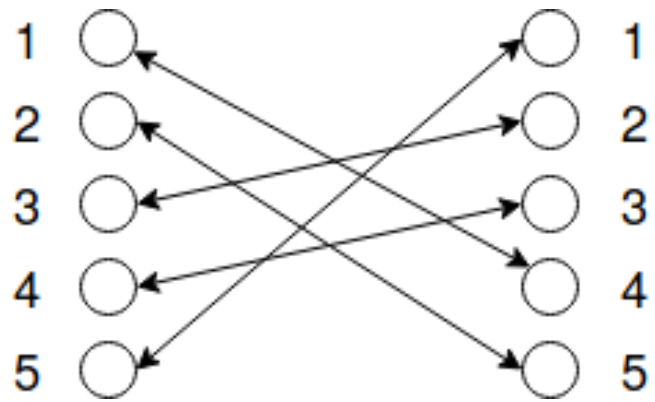
Задача хорошо изучена.
В 1955 Кун опубликовал решение ее решение в виде Венгерского алгоритма
В 1957 Манкрес определил, что алгорим является строго полиномианльным
Карп улучшил его, добившись временной сложности $O(n^3)$

Понятие перестановки

Будем понимать под перестановкой упорядоченный набор без повторений чисел $1, 2, \ldots n$, то есть биекцию на множестве $1, 2, \ldots n$

Пусть $n = 5$, тогда одной из возможных перестановок является

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



Обратная перестановка

Для любой перестановки $\varphi \in S_n$ сущесвует обратная перестановка $\varphi^{-1} \in S_n$, такая что $\varphi \circ \varphi^{-1} = \varphi^{-1} \circ \varphi = e$. Она будет иметь вид

$$\varphi^{-1} = \begin{pmatrix} \varphi(1) & \varphi(2) & \dots & \varphi(n) \\ 1 & 2 & \dots & n \end{pmatrix} \tag{6}$$

Понятие назначения

Мы можем представлять назначение как некое биективное отображение φ, которое ставит элементы конечного множества U в соотвествие элементам конечного множества V.

Связь перестановок и назначение

назначение является перестановкой, которая записывается в виде

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

Матрица назначений

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

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

Матрицу X будем называть матрицей назначений

Естественное обобщение ЗОН

Рассмотрим обобщение ЗОН – трехиндексную аксиальную задачу о назначениях
Она может быть определена следующим образом. Пусть даны n^3 весовых коэфициентов $c_{ijk}, (i, j, k = 1 \dots n)$. 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 $1 \dots n$.

3-АЗОН как ЗЛП

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

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

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

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

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

Известные алгоритмы

- Точный метод ветвей и границ, упорядочивающий полный перебор
- Алогритмы, выделяющие классы решений из $n-$ мерного многогранника всех возможных решений
- Подходы для получения оценок точного решения
- Различные приближенные алгоритмы

Алгоритм, сводящий 3-АЗОН к двухмерному виду

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 $f_{\mathrm{A}(\phi_n)}$.

Теоретическое обоснование

Пусть весовые коэфициенты $c_{ijk} \in C$ лежат в отрезке $[a_n, b_n]$, $a_n > 0$, M_n – множество всех возможных C. Тогда

Теорема

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

Теорема

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

Выявленные недостатки. Предлагаемые модификации

- Алгоритм чувствителен к выбору начальной перестановки
- Показна сходимость алгоритма при $n \to \infty$, что неприменимо в практических задачах
- Для матрицы, полученной после уменьшения размерности, предлагается использовать алгоритм, не гарантирующий лучшее решение.

Для исправления этих недостатков предлагаются следующие модификации к алгоритму

Наивный итеративный алгоритм

1. Создадим счетчик q, изменяющийся от 1 до P
2. $f_{\text{окончательное}} = M$
3. Запустим исходный алгоритм, получим f
4. $f_{\text{окончательное}} = \min\{f, \text{окончательное}\}$
5. $q = q + 1$, если $q < p$, то идем на шаг 3
6. Получено решение $f_{\text{окончательное}}$

Улучшенный итеративный алгоритм

1. Создадим счетчик q, изменяющийся от 1 до P
2. $f_{\text{окончательное}} = M$, π – случайная перестановка
3. Запустим исходный алгоритм с перестановкой π, получим f
4. Если $f_{\text{окончательное}} < f$, то
   1. $f_{\text{окончательное}} = f$
   2. $g \neq h$ – два случайных числа, $\pi[g], \pi[h] = \pi[h], \pi[g]$ иначе π – случайная перестановка
5. $q = q + 1$, если $q < p$, то идем на шаг 3
6. получено решение $f_{\text{окончательное}}$

Модификация с использованием алгоритма поиска точного решения классичесой ЗОН

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 $f = 0$
2. Запустим Венгерский алгоритм на матрице d
3. Результатом работы алгоритма $\mathrm{A}(\phi_n)$ является значение функции f целевой функции $f_{\mathrm{A}(\phi_n)}$.

Модификация целочисленной функции ϕ_n

Утверждается, что для любой целочисленной ϕ_n алгоритм сходится. Сложность алгоритма зависит от ϕ_n и равна $o(n\phi_n(n))$

Обзор программного комплекса

Эксперимент выполнен на языке python3.6

1. Модуль Premutation
2. Модуль Test
3. Модуль Algorithm
4. Модуль ReportAggregation

Численный эксперимент

Входные данные эксперимента:

- Матрицы различной размерности
- Матрицы, заполненные по различным правилам

Виды матриц

Матрицы вида C_{rand}, сгенерированные случайным образом из $\{1 \dots n\}$
Матрицы вида C_{rand}, которые представляют собой матрицу $c \in C_{rand}$, для которой выполнено $c_{ijk} = 0, \quad i, j, k = 1, \dots, n$

Виды матриц

Матрицы специального вида C_{id}, строящиеся по следующему правилу:

$$c_{ijk} = \begin{cases} 0, & \text{i} = \text{j} = \text{k} \\ 1, & \text{иначе,} \end{cases}, \quad i, j, k = 1, \ldots, n \tag{7}$$

О методе оценивания

Предлагается оценивать результат алгоритма и его сходимость через нормализованный вывод

$$f_{normalized} = \frac{f}{n}, \tag{8}$$

где f – вывод алгоритма f на матрице размерности n.
Ожидается, что для выводов $f^{(n_1)}$, $f^{(n_2)}$ и $f^{(n_3)}$, где $n_3 >> n_2 >> n_1$ значение $\Delta f_{3-2} = f^{(n_3)} - f^{(n_2)}$ будет меньше, чем $\Delta f_{2-1} = f^{(n_2)} - f^{(n_1)}$:

$$\Delta f_{3-2} < \Delta f_{2-1} \tag{9}$$

Результаты рассчета

- Алгоритм 2.1 – исходный алгоритм
- Алгоритм 2.2 – наивный итеративный алгоритм
- Алгоритм 2.3 – улучшенный итеративный алгоритм
- Алгоритм 2.4 – алгоритм с использованием венгерского метода

алгоритм$\vert n$	10	20	50	100	150	500	1000	1250	1500	1750
Алгоритм 2.1	14.61	35.48	103.92	220.28	334.79	1126.03	2253.39	2818.79	3376.94	3946.76
Алгоритм 2.2	14.66	31.32	91.66	194.95	299.87	1043.2	2125.39	2665.64	–	–
Алгоритм 2.3	14.42	33.51	94.58	198.18	511.22	1047.52	–	–	–	–
Алгоритм 2.4	21	22	50	100	150	501	–	–	–	–

Вывод программы для основного алгоритма со случайно сгенерированными матрицами

n	f	$f_{normalized}$	$\Delta f_{normalized}$
10	14.61	1.4610	–
20	35.48	1.7740	0.3130
50	103.92	2.0780	0.3039
100	220.28	2.2020	0.1240
150	334.79	2.2320	0.0300
500	1126.03	2.2520	0.0199
1000	2253.39	2.2530	0.0010
1250	2818.79	2.2550	0.0019
1500	3376.94	2.2552	0.0001
1750	3946.76	2.2552	0.0

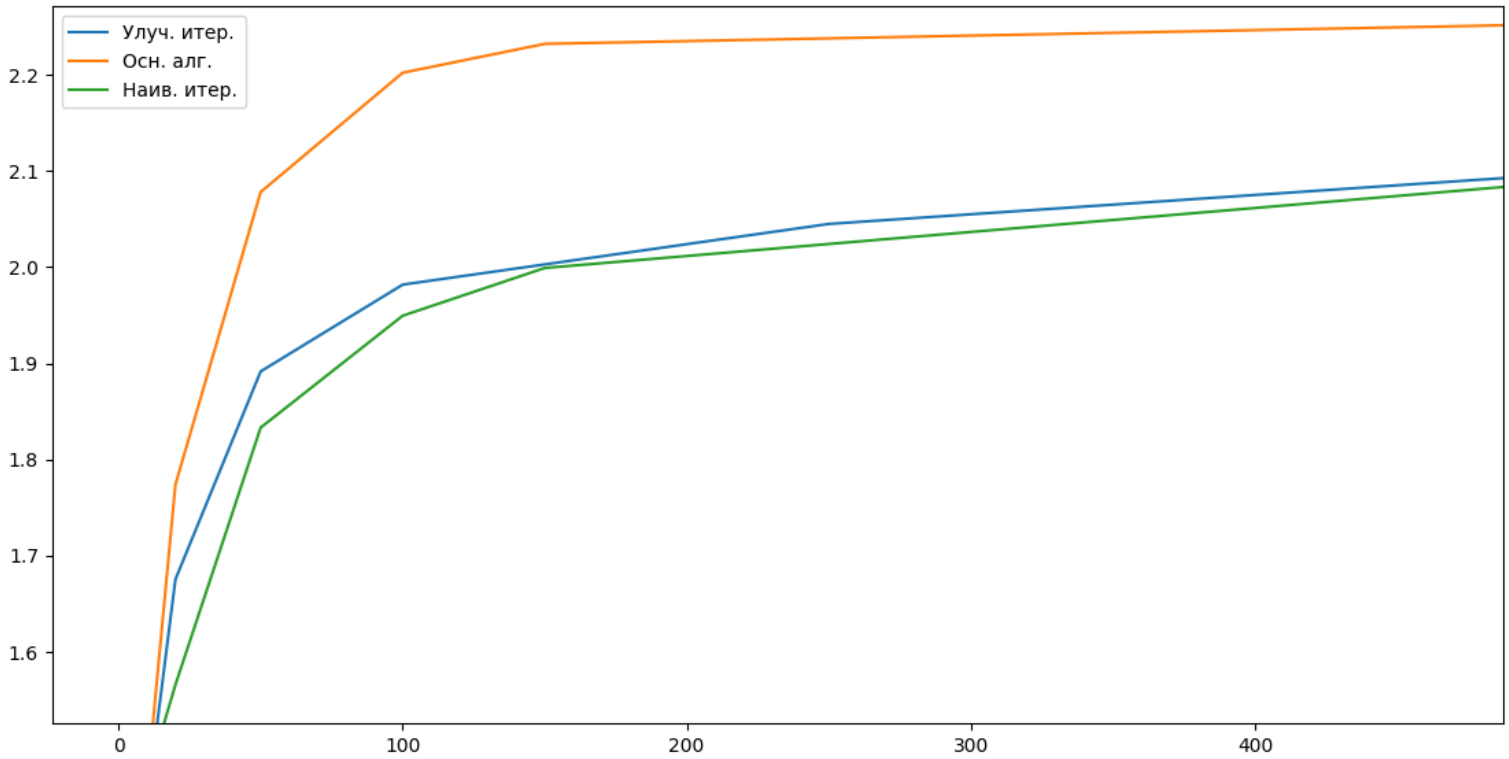
Вывод программы для наивного итеративного алгоритма со случайно сгенерированными матрицами

n	f	$f_{normalized}$	$\Delta f_{normalized}$
10	14.66	1.4660	–
20	31.32	1.5660	0.1000
50	91.66	1.8332	0.2672
100	194.95	1.9495	0.1163
150	299.87	1.9991	0.0496
500	1043.2	2.0864	0.0873
1000	2125.39	2.1254	0.0390
1250	2665.64	2.1325	0.0071

Вывод программы для улучшенного итеративного алгоритма со случайно сгенерированными матрицами

n	f	$f_{normalized}$	$\Delta f_{normalized}$
10	14.42	1.442	–
20	33.51	1.6755	0.2330
50	94.58	1.8916	0.2161
100	198.18	1.9818	0.0902
250	511.22	2.0448	0.06308
500	1047.52	2.0950	0.05024

Результат для матриц из C_{rand}



Вывод программы для основного алгоритма со случайно сгенерированными матрицами с нулевой диагональю

n	f	$f_{normalized}$	$\Delta f_{normalized}$
10	14.08	1.4080	–
20	35.84	1.7920	0.3840
50	103.48	2.0696	0.2776
100	216.12	2.1612	0.0916
250	552.48	2.20992	0.0487
500	1133.76	2.26052	0.0506
750	1697.91	2.26388	0.0034
1000	2261.52	2.26152	-0.0024

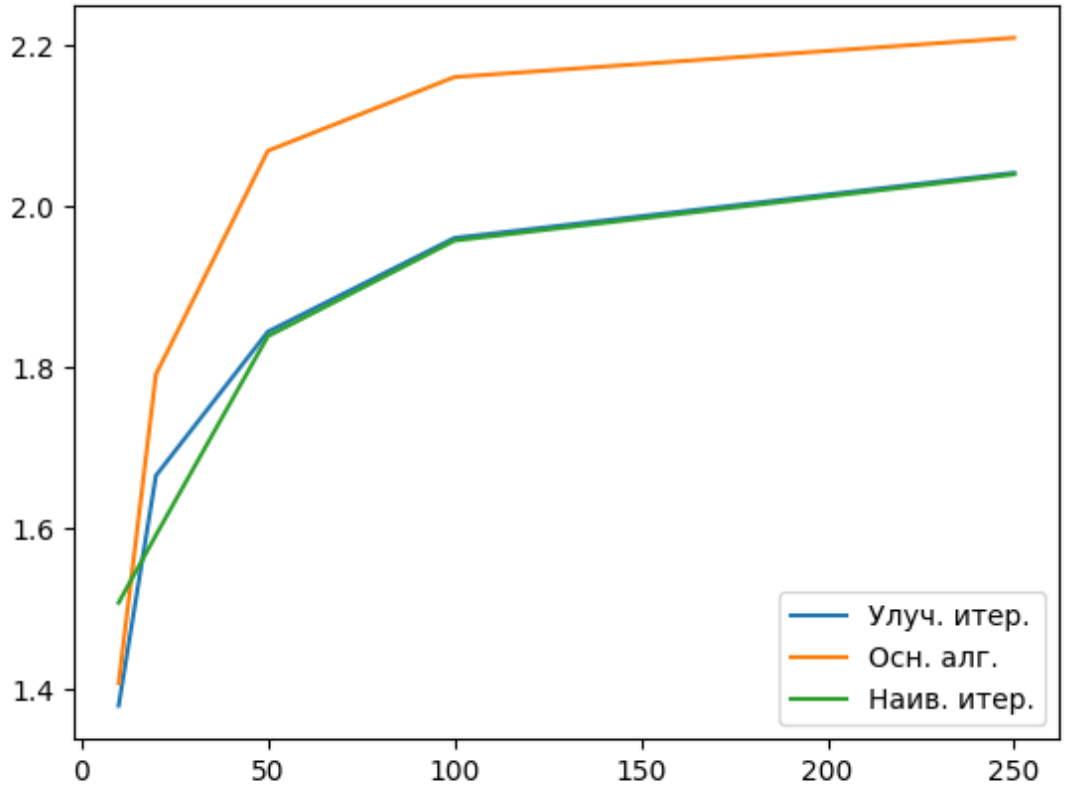
Вывод программы для наивного итеративного алгоритма со случайно сгенерированными матрицами с нулевой диагональю

n	f	$f_{normalized}$	$\Delta f_{normalized}$
10	15.08	1.5080	–
20	31.84	1.5920	0.0840
50	91.96	1.8392	0.2472
100	195.8	1.9580	0.1188
250	510.08	2.0403	0.0823

Вывод программы для улучшенного итеративного алгоритма со случайно сгенерированными матрицами с нулевой диагональю

n	f	$f_{normalized}$	$\Delta f_{normalized}$
10	13.80	1.3800	–
20	33.32	1.6660	0.2860
50	92.24	1.8448	0.1788
100	196.12	1.9612	0.1164
250	510.56	2.0422	0.0810

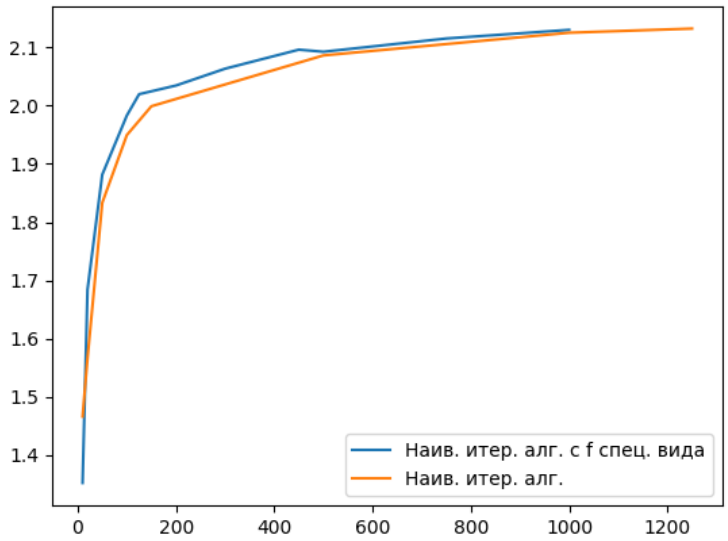
Результат для матриц из C_{rid}



Результат наивного итеративного алгоритма с матрицами из C_{rid} с различными ϕ_n

$$\phi_n(i) = i \tag{10}$$

$$\phi_n(i) = i \bmod 3 \tag{11}$$



Вывод программы для алгоритма с использованием венгерского м., $c \in C_{rand}$

n	f	$f_{normalized}$	$\Delta f_{normalized}$
10	21	2.1	–
20	22	1.1	1.000
50	50	1.0	0.000
100	100	1.0	0.000
250	250	1.0	0.000

Вывод программы для алгоритма с использованием венгерского м., $c \in C_{rid}$

n	f	$f_{normalized}$	$\Delta f_{normalized}$
10	16	1.6	–
20	23	1.15	-0.4500
50	48	0.96	-0.1900
100	98	0.98	0.0200
250	248	0.992	0.0120

Вывод программы для алгоритма с использованием венгерского м., $c \in C_{id}$

n	f	$f_{normalized}$	$\Delta f_{normalized}$
10	16	1.6	–
20	23	1.15	-0.4500
50	48	0.96	-0.1899
100	98	0.98	0.0200
250	248	0.99	0.0120

Вывод

1. исходный алгорим 2.1 асимптотически сходится, теория подтверждена;
2. предложенные модификации (Алгоритмы 2.2, 2.3, 2.4) асимптотически сходятся, причем сходятся быстрее исходного алгоритма;
3. лучший результат показал алгоритм 2.4 с использованием Венгерского метода.

4. Наивная итеративная модификация (алгоритм 2.2) показала себя хорошо.
5. Улучшенная итеративная модификация также показала себя лучше исходного алгоритма, однако, не превосходит наивную итеративную модификацию
6. Модификация с использованием Венгерского м., показал лучший результат, но он потребляет много памяти

Спасибо за внимание