Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Базовая кафедра «Интеллектуальные системы управления»

УТВЕРЖДАЮ
Заведующий кафедрой

_____ _____
подпись инициалы, фамилия
« ____» _____ 20__ г.

БАКАЛАВРСКАЯ РАБОТА

27.03.03 «Системный анализ и управление»

Алгоритм решения расширенной задачи о назначениях

Руководитель _____ доцент, канд. техн. наук А.А. Даничев_
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ Д.А. Колосков
подпись, дата инициалы, фамилия

Красноярск 2020

РЕФЕРАТ

Выпускная квалификационная работа по теме «Алгоритм решения расширенной задачи о назначениях» содержит 29 страниц текстового документа, 6 использованных источников, 16 иллюстраций, 21 формулу.

Объект исследования – задача о распределении сотрудников по помещениям.

Цель исследования – разработка алгоритма решения задачи о распределении сотрудников по помещениям.

В результате проведения были изучены литературные источники, выявлены методы решения задачи о назначениях, а также составлены алгоритмы и проведена апробация на скорость обработки согласованных матриц разных размеров с условиями ограничений по распределению сотрудников по рабочим местам каждого помещения.

В итоге был разработан алгоритм решения расширенной задачи о назначениях, который выявляет оптимальное решение за короткое время, в числе и для задач большой размерности.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Исключительные темпы технического прогресса и усложнение хозяйственных связей породили проблему создания систем управления сложными системами, которая, в свою очередь, привела к необходимости построения математических моделей принятия оптимальных решений.

Задача о назначениях – вид задачи линейного программирования, с помощью которой решаются вопросы типа: как распределить рабочих по станкам, чтобы общая выработка была наибольшей или затраты на заработную плату наименьшими (поскольку для каждой комбинации «рабочий — станок» характерна своя производительность труда), как наилучшим образом распределить экипажи самолетов, как назначить людей на различные должности

Актуальность. 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[1].

На практике возникают различные вариации задачи о назначениях. Например, известны обобщенная задача о назначениях, задача о назначении минимального количества исполнителей, линейная задача о назначениях в

узких местах, квадратичная задача о назначениях, задача о марьяже, задача о соседях по комнате, задача о назначении целей. В данной работе **объектом исследования** является задача о распределении сотрудников по помещениям. Предприятие набирает рабочий персонал. Каждый рабочий имеет свой уровень адаптированности (предпочтения) к условиям предоставленных помещений. Необходимо найти такой вариант распределения, который бы максимизировал суммарный уровень адаптированности.

Предмет исследования: методы линейного целочисленного математического программирования.

Цель работы: разработка алгоритма решения задачи о распределении сотрудников по помещениям.

Для реализации цели выпускной квалификационной работы были поставлены **следующие задачи**:

- подбор и анализ литературных источников;
- изучение алгоритмов решения задачи о назначениях;
- формализация задачи о распределении сотрудников по помещениям;
- разработка алгоритма решения задачи;
- программная реализация алгоритма и его апробация.

1 Теоретическая часть

1.1 Задача о назначениях

Частным случаем транспортной задачи является задача о назначениях, в которой число пунктов производства равно числу пунктов назначения, иначе говоря, транспортная таблица имеет форму квадрата. Кроме того, в каждом пункте назначения объем потребности равен 1, и величина предложения каждого пункта производства равна 1. Любая задача о назначениях может быть решена с использованием методов линейного программирования или алгоритма решения транспортной задачи. Однако ввиду особой структуры данной задачи был разработан специальный алгоритм, получивший название Венгерского метода[2].

Варианты задачи о назначениях представлены в таблице 1.

Таблица 1 – Варианты задачи о назначениях

Ресурсы	Потребители (объекты)	Критерий эффективности
Исполнители	Работы	Время выполнения (мин), профессионал. соответствие (баллы)
Автомобили	Маршруты	Объём перевозимой продукции
Станки	Работа (участки)	Минимальное время или максимальная производительность

Оптимальный подбор назначений должен достигать за счет максимизации или минимизации определенной меры эффективности назначения: прибыли или стоимости. Для каждого потенциального назначения оценивается мера эффективности. Если мерой эффективности является прибыль, то в процессе решения задачи о назначениях она максимизируется, если мерой эффективности является стоимость, она минимизируется[2].

Выполним математическую постановку сбалансированной задачи о назначениях, для чего:

- Определим неизвестные и их количество. Рассмотрим переменные x_{ij}, которые равны 1, если *i*-й исполнитель назначен на выполнение *j*-ой работы и 0, если он не назначен *i*=1,2,…,n, *j*=1,2,…,n. Таким образом, значения x_{ij} образуют матрицу назначений $X_{n\times n}$, состоящую из нулей и единиц;

- Запишем критерий оптимизации (целевую функцию) – суммарную эффективность (неэффективность) выполнения всех работ. Целевая функция задачи о назначениях зависит от неизвестных пока переменных x_{ij} и известных значений c_{ij} – эффективности (неэффективности) выполнения *i*-м исполнителем *j*-ой работы и запишется в следующем виде:

$$F(X) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} \cdot x_{ij} \to \max\,(-min) \qquad (1.1.1)$$

-

- Сформулируем ограничения рассматриваемой задачи. Делается это из нескольких шагов:

1) каждый исполнитель выполняет только одну работу. Выполнение данного условия означает, что каждая строка матрицы назначений X содержит только одно значение равное единицы, а все остальные равны нулю. Т.е. сумма элементов каждой строки матрицы назначений *X* равна 1:

$$\sum_{j=1}^{m} x_{ij} = 1,\ \ i = \overline{1,n} \qquad (1.1.2)$$

2) каждая работа выполняется только одним исполнителем. Выполнение данного условия означает, что каждый столбец матрицы назначений X содержит только одно значение равное единицы, а все остальные равны нулю. Т.е. сумма элементов каждого столбца матрицы X равна 1:

$$\sum_{i=1}^{n} x_{ij} = 1, \;\; j = \overline{1,m} \qquad (1.1.3)$$

3) двоичность (бинарность) переменных x_{ij}, т.к. областью допустимых изменений каждой переменной является не множество целых неотрицательных чисел, а конечное множество (0,1)

$$x_{ij} = \begin{cases} 1, & \text{если } i-\text{й исполнитель назначен на } j \text{ работу} \\ 0, & \text{если } i-\text{й исполнитель не назначен на } j \text{ работу} \end{cases} \qquad (1.1.4)$$

то мы получаем дискретную задачу оптимизации[3].

Таким образом, целевая функция (1.1.1) и ограничения (1.1.2−1.1.4) составляют математическую модель сбалансированной задачи о назначениях.

Не сбалансированная задача о назначениях, когда число исполнителей n не равно числу выполняемых работ m. При этом возможны два случая:

- число исполнителей n больше числа выполняемых работ m ($n > m$);
- число исполнителей n меньше числа выполняемых работ m ($n < m$).

В обоих случаях для решения не сбалансированной задачи ее сводят к сбалансированной, путем введения фиктивных работ или фиктивных исполнителей с нулевыми значениями c_{ij} – эффективности (неэффективности) выполнения i-м исполнителем j-ой работы. В первом случае, когда n > m, вводится k = n – m фиктивных работ $Р_{m+1}$, $Р_{m+2}$, …, $Р_{m+k}$. Во втором случае (n < m) – рассматриваются k = m – n фиктивных исполнителя $И_{n+1}$,$И_{n+2}$, …, $И_{n+k}$[3].

1.2 Методы решения задачи о назначениях

Экономические задачи по своей природе являются задачами выбора. Общепринято, что этот выбор нацелен на т.н. оптимальное решение, т.е. на максимальный или минимальный результат. Существующие математические

методы решения многих экономических задач преследуют ту цель, чтобы сразу найти оптимальное решение, т.е. найти максимум/минимум итерационными методами, когда очередной план решения проверяется каким-либо способом, является ли он оптимальным. И, если нет, то с помощью определенных приемов это решение меняется в сторону улучшения, после чего все повторяется до тех пор, пока очередной план решения не будет признан оптимальным.

1.2.1 Метод полного перебора

Под полным перебором понимается методика разрешения задач математики путем рассмотрения всех возможных вариантов. Уровень сложности при полном переборе напрямую связан с количеством допустимых решений задачи. В случае, когда область решений огромна, время полного перебора может исчисляться десятками и даже сотнями лет, и при этом итоговый результат возможно ещё не будет найден. Все задачи класса NP (non-deterministic polynomial) могут быть решены при помощи полного перебора. В криптографии оценивается криптографическая стойкость шифрования на базе вычислительной сложности полного перебора[2]. Например, криптостойкость шрифта будет на должном уровне, если нет методики его взлома, которая позволяет это сделать быстрее, чем путём полного перебора всего диапазона возможных ключей. Хакерские атаки в области криптографии, которые основаны на методике полного перебора, считаются наиболее универсальными, но при этом требуют максимальных временных затрат.

1.2.2 Венгерский метод

Идея метода была высказана венгерским математиком Эгервари и состоит в следующем. Строится начальный план перевозок, не удовлетворяющий в общем случае всем условиям задачи (из некоторых пунктов производства не

весь продукт вывозится, потребность части пунктов потребления не полностью удовлетворена). Далее осуществляется переход к новому плану, более близкому к оптимальному. Последовательное применение этого приема за конечное число итераций приводит к решению задачи.

Алгоритм состоит из предварительного этапа и не более чем (n-2) последовательно проводимых итераций. Каждая итерация связана с эквивалентными преобразованиями матрицы, полученной в результате проведения предыдущей итерации, и с выбором максимального числа независимых нулей. Окончательным результатом итерации является увеличение числа независимых нулей на единицу. Как только количество независимых нулей станет равным n, проблему выбора оказывается решенной, а оптимальный вариант назначений определяется позициями независимых нулей в последней матрице[3].

Если задача решается на определение максимума, то алгоритм выглядит следующим образом:

Предварительный этап. Разыскивают максимальный элемент в j-м столбце и все элементы этого столбца последовательно вычитают из максимального. Эту операцию проделывают над всеми столбцами матрицы C. В результате образуется матрица с неотрицательными элементами, в каждом столбце которой имеется, по крайней мере, один нуль.

Далее рассматривают i-ю строку полученной матрицы, разыскивают ее минимальный элемент и из каждого элемента этой строки вычитают минимальный. Эту процедуру повторяют со всеми строками. В результате получим матрицу C_0 ($C_0 \sim C$), в каждой строке и столбце которой имеется, по крайней мере, один нуль. Описанный процесс преобразования C в C_0 называется приведением матрицы.

Находим произвольный нуль в первом столбце и отмечаем его звездочкой. Затем просматриваем второй столбец, и если в нем есть нуль, расположенный в строке, где нет нуля со звездочкой, то отмечаем его звездочкой. Аналогично просматриваем один за другим все столбцы матрицы

C_0 и отмечаем, если возможно, следующие нули знаком '*'. Очевидно, что нули матрицы C_0, отмеченные звездочкой, являются независимыми. На этом предварительный этап заканчивается.

Стоит учесть, что если в матрице C_k k-й итерации имеется ровно n нулей со звездочкой, то процесс решения заканчивается. В противном случае переходим к (k+1) - й итерации.

Каждая итерация начинается первым и заканчивается вторым этапом. Между ними может несколько раз проводиться пара этапов: третий - первый. Перед началом итерации знаком '+' выделяют столбцы матрицы C_k, которые содержат нули со звездочками.

Первый этап. Просматривают невыделенные столбцы C_k. Если среди них не окажется нулевых элементов, то переходят к третьему этапу. Если же невыделенный нуль матрицы C_k обнаружен, то возможен один из двух случаев:

- строка, содержащая невыделенный нуль, содержит также и нуль со звездочкой;

- эта же строка не содержит нуля со звездочкой.

Во втором случае переходим сразу ко второму этапу, отметив этот нуль штрихом. В первом случае этот невыделенный нуль отмечают штрихом и выделяют строку, в которой он содержится (знаком '+' справа от строки). Просматривают эту строку, находят нуль со звездочкой и уничтожают знак '+' выделения столбца, в котором содержится данный нуль.[3]

Далее просматривают этот столбец (который уже стал невыделенным) и отыскивают в нем невыделенный нуль (или нули), в котором он находится. Этот нуль отмечают штрихом и выделяют строку, содержащую такой нуль (или нули). Затем просматривают эту строку, отыскивая в ней нуль со звездочкой.

Этот процесс за конечное число шагов заканчивается одним из следующих исходов:

- все нули матрицы C_k выделены, т.е. находятся в выделенных строках или столбцах (в этом случае они переходят к третьему этапу);

- имеется такой невыделенный нуль в строке, где нет нуля со звездочкой (в этом случае нули переходят ко второму этапу, отметив этот нуль штрихом).

Второй этап. На этом этапе строят следующую цепочку из нулей матрицы C_k: исходный нуль со штрихом, нуль со звездочкой, расположенный в одном столбце с первым нулем со штрихом в одной строке с предшествующим нулем со звездочкой и т.д. Итак, цепочка образуется передвижением от 0' к 0* по столбцу, от 0* к 0' по строке и т.д. Можно доказать, что описанный алгоритм построения цепочки однозначен и конечен, при этом цепочка всегда начинается и заканчивается нулем со штрихом. Далее над элементами цепочки, стоящими на нечетных местах (0'), ставим звездочки, уничтожая их над четными элементами (0*). Затем уничтожаем все штрихи над элементами C_k и знаки выделения '+'. Количество независимых нулей будет увеличено на единицу. На этом (k+1) -я итерация закончена[3].

Третий этап. К этому этапу переходят после первого, если все нули матрицы C_k выделены. В таком случае среди невыделенных элементов C_k выбирают минимальный и обозначают его h (h>0). Далее вычитают h из всех элементов матрицы C_k, расположенных в невыделенных строках и прибавляют ко всем элементам, расположенным в выделенных столбцах. В результате получают новую матрицу C'_k, эквивалентную C_k. Заметим, что при таком преобразовании, все нули со звездочкой матрицы C_k остаются нулями и в C'_k, кроме того, в ней появляются новые невыделенные нули. Поэтому переходят вновь к первому этапу. Завершив первый этап, в зависимости от его результата либо переходят ко второму этапу, либо вновь возвращаются к третьему этапу. После конечного числа повторений очередной первый этап обязательно закончится переходом на второй этап. После его выполнения количество независимых нулей увеличится на единицу и (k+1) - я итерация будет закончена[3].

1.3 Транспортная задача

Транспортная задача – задача о наиболее рациональном плане перевозок однородного продукта из пунктов производства в пункты потребления. Пусть имеется m пунктов производства некоего однородного продукта A_1, …, A_i, …, A_n и _m пунктов его потребления B_1, …, B_i, …, B_m. В пункте A_i $(i = 1, \dots, n)$ производится a_i единиц, а в пункте B_j $(j = 1, \dots, m)$ потребляется b_j единиц продукта. Предполагают, что:

$$\sum_{i=1}^{n} a_i = \sum_{j=1}^{m} b_j \tag{1.3.1}$$

Транспортные издержки, связанные с перевозкой единицы продукта из пункта A_i в пункт B_j равны c_{ij}. Суть транспортной задачи состоит в составлении оптимального плана перевозок, минимизирующего суммарные транспортные издержки и при реализации которого запросы всех пунктов потребления B_j $(j = 1, \dots, m)$, были бы удовлетворены за счет производства продукта в пунктах A_i $(i = 1, \dots, n)$[3]. Пусть x_{ij} количество продукта, перевозимого из пункта A_i в пункт B_j. Тогда транспортная задача математически формулируется так: определить значения переменных x_{ij} $(i = 1, \dots, n;\ j = 1, \dots, m)$, минимизирующих суммарные транспортные издержки:

$$Z(x) = \sum_{i=1}^{n} \sum_{j=1}^{m} (c_{ij} x_{ij}) \to min \tag{1.3.2}$$

при условиях

$$a_i = \sum_{j=1}^{m} x_{ij}\,, \qquad i = 1, \dots, n \tag{1.3.3}$$

$$b_i = \sum_{i=1}^{n} x_{ij}\,, \qquad j = 1, \dots, m \tag{1.3.4}$$

$$x_{ij} \geq 0, \qquad i = 1, \dots, n;\ j = 1, \dots, m \tag{1.3.5}$$

Набор чисел x_{ij} $(i = 1, \dots, n;\ j = 1, \dots, m)$, который удовлетворяет этим условиям, называется планом перевозок, а его элементы – перевозками. План перевозок, который минимизирует суммарные транспортные издержки, называется оптимальным[3].

Пусть P_{ij} – это (n + m)-мерный вектор, i-я и (n + j)-я компоненты которого равны 1, а остальные составляющие – 0. План перевозок называется опорным, если система векторов P_{ij}, соответствующих положительным перевозкам x_{ij}, линейно независима. Если опорный план перевозок содержит (n + m - 1) положительных перевозок, то он невырожденный. В противном случае имеет место вырожденность опорного плана перевозок. Транспортная задача называется невырожденной, если все ее опорные сланы перевозок невырожденные, а если хотя бы один опорный план перевозок вырожден, то транспортная задача вырождается[3]. Можно доказать, что для невырожденности транспортной задачи необходимо и достаточно, чтобы для любого подмножества пунктов производства $A_{i_1}, A_{i_2}, \dots, A_{i_{k1}}$, не совпадающего со всем множеством пунктов производства, и любого подмножества пунктов потребления $B_{j_1}, B_{j_2}, \dots, B_{j_{k2}}$, выполнялось условие:

$$\sum_{l=1}^{k_1} a_{i_l} \neq \sum_{l=1}^{k_2} b_{j_l} \tag{1.3.6}$$

Для устранения вырожденности, транспортная задача незначительно изменяется, в результате получается новая невырожденная. В новой транспортной задаче объемы производств $\tilde{a}_i$ в пунктах A_i $(i = 1, \dots, n)$ равны:

$$\tilde{a}_i = a_i + \varepsilon, \tag{1.3.7}$$

а объемы потребления $\widetilde{b_j}$ пунктов B_j $(j = 1, \dots, m)$, равны:

$$\widetilde{b_j} = \begin{cases} b_j, \ j = 1, \dots, m-1 \\ b_j + n * \varepsilon, \ j = m \end{cases}, \tag{1.3.8}$$

где

$$0 < \varepsilon < \frac{1}{n-1} \tag{1.3.9}$$

При достаточно малом ε решение новой транспортной задачи близко к решению исходной транспортной задачи, причем новая транспортная задача невырожденная. Последовательность коммуникаций $(A_{i_1}, B_{j_1}), (A_{i_2}, B_{j_1}), (A_{i_2}, B_{j_2}), \dots, (A_{i_{s-1}}, B_{j_{s-1}}), (A_{i_s}, B_{j_{s-1}}), (A_{i_s}, B_{j_s})$, называется цепочкой, связывающей пункты A_{i_1} и B_{j_s}, (A_i, B_j) – коммуникация (дорога), связывающая пункт производства A_i с пунктом потребления B_j. Если к этой цепочке добавить коммуникацию (A_{i_s}, B_{j_s}), то получим замкнутую цепочку[3].

Для решения классических транспортных задач, в основном, применяется метод потенциалов.

Метод потенциалов основан на условиях оптимальности плана перевозок, которые формулируются так. Для оптимальности данного плана перевозок x_{ij} $(i = 1, \dots, n; \ j = 1, \dots, m)$, необходимо и достаточно существование чисел

u_i $(i = 1, \dots, n)$, и v_j $(j = 1, \dots, m)$, называемых потенциалами, таких, что выполняются следующие условия:

$$v_j - u_i \leq c_{ij}, \text{ если } x_{ij} = 0; \quad (1.3.10)$$

$$v_j - u_i = c_{ij}, \text{ если } x_{ij} > 0; \quad (1.3.11)$$

Этот метод дает возможность, отправляясь от некоторого невырожденного опорного плана перевозок, построить за конечное число итераций опорный план перевозок, также невырожденный, являющийся решением транспортной задачи. Отдельная итерация метода заключается в преобразовании невырожденного опорного плана перевозок, полученного на предыдущей итерации т. о., что в результате получается новый невырожденный, опорный план перевозок, связанный с меньшими суммарными транспортными издержками. Преобразование опорного плана перевозок осуществляется с помощью некоторой замкнутой цепочки. На каждой итерации метода потенциалов требуется невырожденность опорного плана перевозок. Это достигается применением метода устранения вырожденности транспортной задачи[4].

2 Практическая часть

2.1 Постановка задачи

Предприятие набирает рабочий персонал. Каждый рабочий имеет свой уровень адаптированности (предпочтения) к условиям предоставленных помещений.

Необходимо найти такой вариант распределения, который бы максимизировал суммарный уровень адаптированности. Для каждого помещения указаны минимально и максимально возможное количество рабочих.

Например, дано: 5 претендентов и 2 помещения, в каждом из которых имеется ограничения по необходимому числу рабочих, т.е. в 1-ом – от 1-го до 2-х, во 2-ом – от 3-х до 5-х. Необходимо распределить рабочих по помещениям таким образом, чтобы они приносили больше пользы предприятию.

В таблице 2 представлены уровни предпочтения сотрудников по помещениям.

Таблица 2 – Уровни предпочтения

	Помещения	
	1	2
Сотрудник 1	4	5
Сотрудник 2	9	7
Сотрудник 3	1	2
Сотрудник 4	4	3
Сотрудник 5	3	0

На рисунке 1 представлена матрица с уровнями предпочтения и двудольный граф с соответствующими весами.

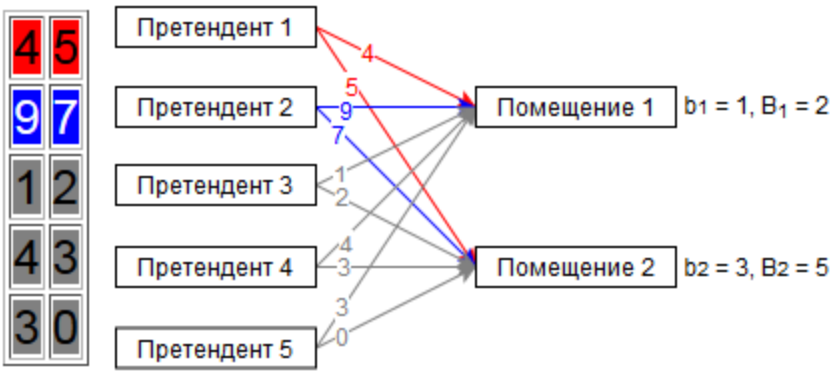

Рисунок 1 – Наглядное представление задачи двудольным графом

На рисунке 2 представлен двудольный граф решения задачи о назначениях.

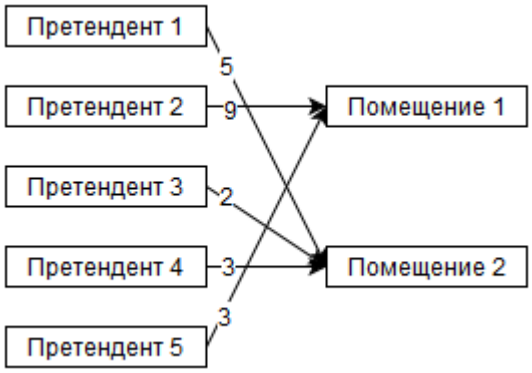

Рисунок 2 – Результат решения задачи о назначениях

2.2 Математическая постановка задачи

Выполним математическую постановку задачи, для чего:

- Определим неизвестные и их количество. Рассмотрим переменные x_{ij}, которые равны 1, если *i*-й сотрудник определен в *j*-ое помещение и 0, если он не определен *i*=1,2,…,n, *j*=1,2,…,m. Таким образом, значения x_{ij} образуют матрицу назначений $X_{n\times m}$, состоящую из нулей и единиц[5];

- Запишем критерий оптимизации (целевую функцию) – суммарную эффективность распределения по помещениям. Целевая функция задачи зависит от неизвестных переменных x_{ij} и известных значений c_{ij} – степень предпочтения i-го сотрудника j-ого помещения и запишется в следующем виде:

$$F(X) = \sum_{i=1,j=1}^{n,m} c_{ij} \cdot x_{ij} \rightarrow \max\ (-min) \tag{2.2.1}$$

- Сформулируем ограничения рассматриваемой задачи[5]. Они будут выглядеть следующим образом:

1) каждый сотрудник должен оказаться в одном помещении:

$$\sum_{j=1}^{m} x_{ij} = 1, i = \overline{1,n} \tag{2.2.2}$$

2) ограничение на количество сотрудников в помещении:

$$b_j \leq \sum_{i=1}^{n} x_{ij}, j = \overline{1,m} \tag{2.2.3}$$

$$\sum_{i=1}^{n} x_{ij} \leq B_j, j = \overline{1,m} \tag{2.2.4}$$

3) двоичность (бинарность) переменных x_{ij}. Так как областью допустимых изменений каждой переменной является не множество целых неотрицательных чисел, а конечное множество (0,1):

$$x_{ij} = \begin{cases} 1, & \text{если } i\text{-й сотрудник определен в } j\text{-ое помещение} \\ 0, & \text{если } i\text{-й сотрудник не определен в } j\text{-ое помещение} \end{cases} \quad (2.2.5)$$

2.3 Преобразование к задаче назначения

Целевая функция 2.2.1 и ограничения 2.2.2 и 2.2.5 соответствуют задачи о назначении 1.1.1-4.

Для приведения задачи к классическому виду выполним ряд действий:

- Так как в классической задачи о назначениях в одно помещение может быть назначен только один работник (условие 1.1.2), то каждое помещение необходимо продублировать B_j раз. Если количество работников не соответствует ёмкости помещений, то необходимо добавить фиктивные помещения или фиктивных работников с нулевыми весами. Размер новой, согласованной матрицы определяется как:

$$N = \max\left(n, \sum_{j=1}^{m} B_j\right) \quad (2.3.1)$$

- Для $j = \overline{1,m}$ в новую матрицу весов $C_{N\times N}$ b_i раз добавляется столбец j из матрицы $c_{n\times m}$. Так как в каждое помещение должно быть назначено не менее b_j работников, то назначение фиктивных работников не допустимо. Все оставшиеся строчки в заполняемых столбцах заполняются -∞;

- Для $j = \overline{1,m}$ в новую матрицу весов $C_{N\times N}$ $(B_j - b_j)$ раз добавляется столбец j из матрицы $c_{n\times m}$. Все оставшиеся строчки в заполняемых столбцах соответствуют фиктивным работникам и заполняются 0[5].

Из примера с раздела 2.1 (см. рисунок 1) выясняется, что n = 5, m = 2, $\sum_{j=1}^{m} B_j = 7$, N = 7. Тогда следующие действия будут таковы:

- Добавить N – n = 2 фиктивных претендента (№6, №7);

- Заполнить столбцы 1-4;

- Заполнить столбцы 5-7.

Ниже представлен рисунок с уже сбалансированной матрицей распределения сотрудников по помещениям с ограничениями.

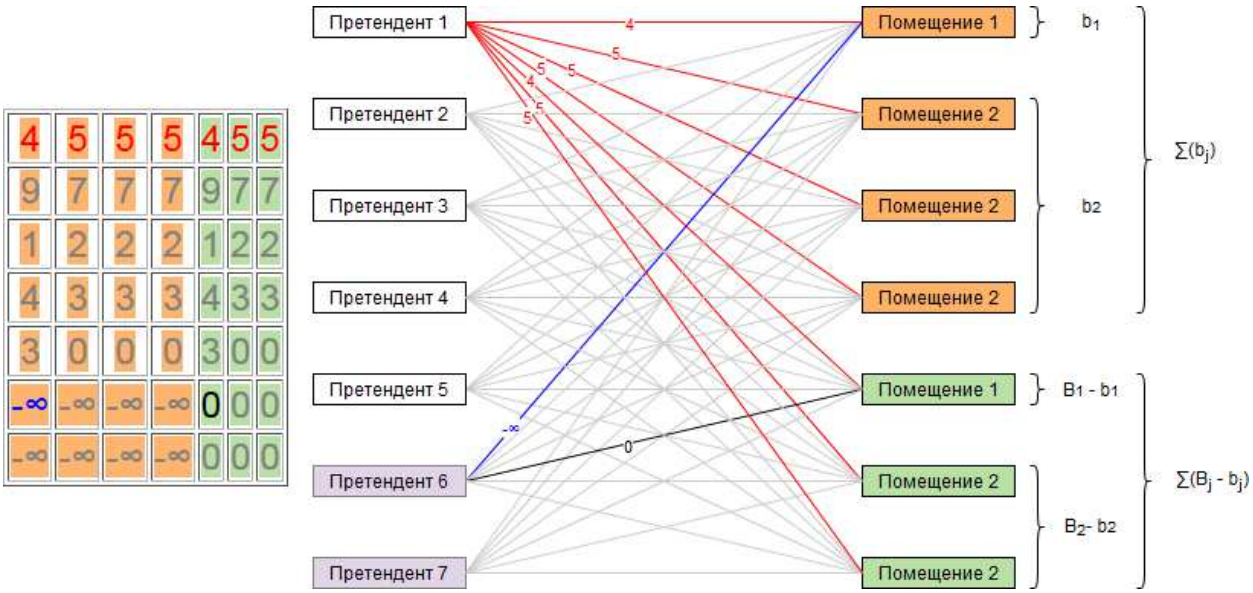

Рисунок 3 – Сбалансированная матрица и двудольный граф классической задачи о назначениях

2.4 Преобразование к транспортной задаче

Для приведения задачи к классическому виду выполним ряд действий. Условия сбалансирования таковы:

- Если сумма помещений больше, чем сумма претендентов, т.е. $\sum_{j=1}^{m} b_j > \sum_{i=1}^{n} a_i$, тогда добавляется строка фиктивного претендента, заполненного нулями, в нижний край матрицы распределения 1 раз, а в колонке наличия претендентов вес будет иметь число, равное $\sum_{j=1}^{m} b_j - \sum_{i=1}^{n} a_i$;

- Если сумма претендентов больше, чем сумма помещений, т.е. $\sum_{i=1}^{n} a_i > \sum_{j=1}^{m} b_j$, тогда добавляется столбец фиктивного помещения, заполненный нулями, в правый край 1 раз, а в строке наличия рабочих мест по помещениям вес будет иметь число, равное $\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j$;

- Если сумма претендентов меньше, чем сумма максимальных норм рабочих мест в помещениях, тогда добавляется строка фиктивного претендента, в которой помимо нулей весы первых $\sum_{j=1}^{m} b_j$ её элементов будут иметь -∞, чтобы фиктивный претендент не претендовал на необходимые рабочие места, а в колонке наличия претендентов вес будет иметь число, равное $\sum_{j=1}^{m} B_j - \sum_{i=1}^{n} a_i$[5].

Из примера с раздела 2.1 (см. рисунок 1) выясняется, что n = 5, m = 2, $\sum_{j=1}^{m} b_j = 4$, $\sum_{j=1}^{m} B_j = 7$. Тогда следующие действия будут таковы:

- Добавить 1 строку с (7 - 5) = 2 фиктивных претендента (№6, №7);

- Заполнить столбцы 3-4.

Ниже представлен рисунок с уже сбалансированной матрицей распределения сотрудников по помещениям с ограничениями.

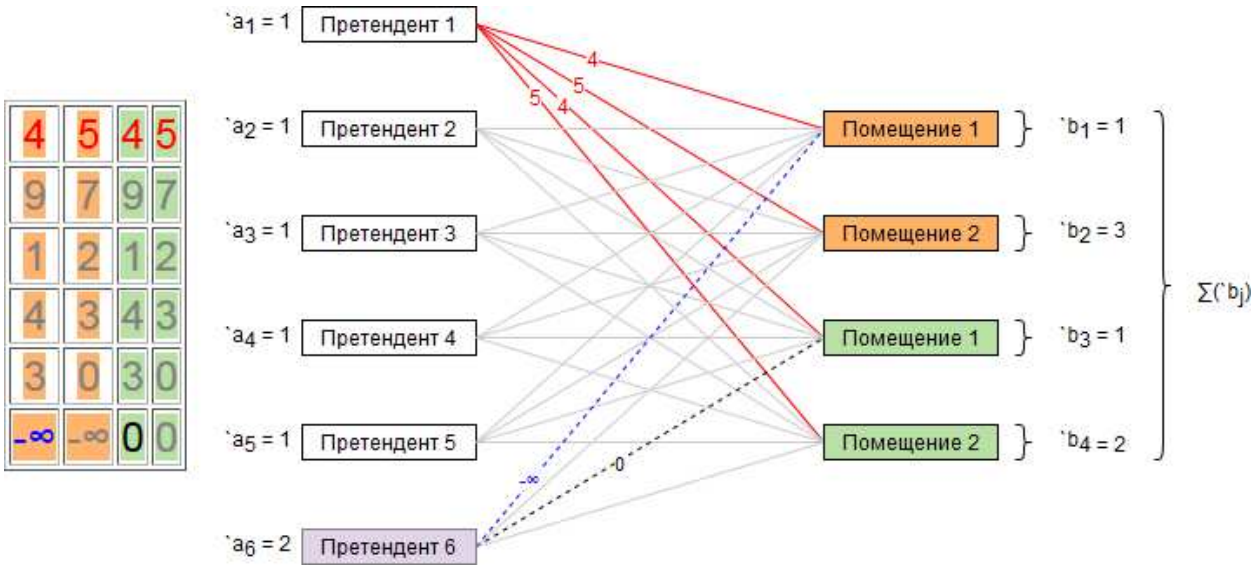

Рисунок 4 – Сбалансированная матрица и двудольный граф классической транспортной задачи

3 Апробация

Апробация разработанного алгоритма решения расширенной задачи о назначениях проводилась серией тестов на синтетических данных. Для фиксированного количества сотрудников и помещений формировались ограничения на количество сотрудников в помещениях, генерировалась матрица весов. Затем применялся один из алгоритмов. Размерность задачи повышалась до тех пор, пока решение находилось в пределах нескольких минут. Для каждой размерности эксперимент выполнялся по 20 раз.

3.1 Программная реализация

Для реализации программных алгоритмов методов решения задачи о назначениях был использован интерпретатор Python с подключёнными библиотеками NumPy, Munkres[6].

На рисунках 5-7 представлены программные реализации алгоритмов отобранных методов решения задачи о назначениях.

```
import numpy;
import random;
import time;

start = time.time();

_array = [];
for i1 in range(5):
    _array.append([]);
    for i2 in range(5):
        _array[i1].append(random.randint(1, 9));

_array_a1 = [];
_array_a2 = [];
_max = len(_array);

def _asd(_array_example):
    dfg = 0;
    for i in range(0, len(_array_example), 1):
        for j in range(0, len(_array_example), 1):
            if i == j: dfg = dfg + int(_array_example[i] == _array_example[j]);
            elif i != j: dfg = dfg + int(_array_example[i] != _array_example[j]);
    return (dfg == _max * _max);
```

Рисунок 5 – Метод полного перебора, лист 1

```
def dfghj(_array, _i):
    _sum = 0;
    for i in range(0, len(_i), 1): _sum += _array[i][_i[i]];
    return _sum;

start = time.time();

for i0 in range(0, len(_array[0]), 1):
    for i1 in range(0, len(_array[1]), 1):
        for i2 in range(0, len(_array[2]), 1):
            for i3 in range(0, len(_array[3]), 1):
                for i4 in range(0, len(_array[4]), 1):
                    if(_asd([i0, i1, i2, i3, i4])):
                        _array_a1.append([i0, i1, i2, i3, i4]);
                        _array_a2.append(dfghj(_array, [i0, i1, i2, i3, i4]));

stop = time.time();
print("Матрица 5x5 обработана за ", round(stop - start, 2), "сек.")
```

Рисунок 5, лист 2

```
import sys;
import time;

from munkres import Munkres, print_matrix, make_cost_matrix

import numpy;
import random;

_time_array = []; # массив временных отметок

_time_array.append(time.time()); # начало


# Дано:

matrix = [[4, 5, 5, 5, 4, 5, 5],
          [9, 7, 7, 7, 9, 7, 7],
          [1, 2, 2, 2, 1, 2, 2],
          [4, 3, 3, 3, 4, 3, 3],
          [3, 0, 0, 0, 3, 0, 0],
          [-90, -90, -90, -90, 0, 0, 0],
          [-90, -90, -90, -90, 0, 0, 0]];


# Вычисление:

# 1. 2 пути алгоритма

cost_matrix = []

for row in matrix:
    cost_row = []

    for col in row:
        cost_row += [sys.maxsize - col]
    cost_matrix += [cost_row]
```

Рисунок 6 – Венгерский метод, лист 1

```
# 2. Обработка

m = Munkres()
indexes = m.compute(cost_matrix)
print_matrix(matrix, msg='Highest profit through this matrix:')
total = 0
for row, column in indexes:
    value = matrix[row][column]
    total += value
    print(f'({row}, {column}) -> {value}')

# 3. Вывод алгоритма
print(f'total profit={total}')

_time_array.append(time.time()); # конец


# Пересчёт времени работы алгоритма
_time_array___length = len(_time_array) - 1;

for i in range(_time_array___length):
    print('Время: ', round(_time_array[i+1] - _time_array[i], 2));
```

Рисунок 6, лист 2

```
import time;
from numpy import *
import lab4

print("Начало программы: ", time.time());

const = 4;
m = 5;
n = 2;

for cvb in range(20): # 20 циклов по одному на каждую случайную матрицу

    start = time.time();

    ___array = [];
    for i1 in range(m):
        ___array.append([]);
        for i2 in range(n):
            ___array[i1].append(random.randint(1, 9));

            C = array(___array);

            ___a = [];
            ___b = [];

            for i in range(m):
                ___a.append(1);
                ___b.append(1);

    a = array(___a);
    b = array(___b);

    lab4.method_of_potentials(C, a, b)

    stop = time.time(); print('Время: ', round(stop - start ,2),'сек.');

    print("Начало программы: ", time.time());
```

Рисунок 7 – Метод потенциалов транспортной задачи

3.2 Метод полного перебора

На таблице 3 изображено времена, за которые методом полного перебора были решены 20 различных матриц, чья размерность – 5x5, 6x6, 7x7, 8x8, 9x9.

Таблица 3 – Результаты времён обработок 20 матриц, решённых методов полного перебора

Кол-во прете-тов	5	6	7	8	9
1	0:00:00,08	0:00:01,69	0:00:39,80	0:17:12,46	8:41:18,82
2	0:00:00,09	0:00:01,41	0:00:40,34	0:18:44,08	8:36:50,93
3	0:00:00,07	0:00:01,46	0:00:38,14	0:18:05,15	9:00:54,06
4	0:00:00,06	0:00:01,68	0:00:42,99	0:18:33,54	8:32:01,62
5	0:00:00,05	0:00:01,63	0:00:37,00	0:20:15,69	9:07:44,32
6	0:00:00,09	0:00:01,59	0:00:38,41	0:19:57,68	8:27:10,06
7	0:00:00,07	0:00:01,52	0:00:38,71	0:19:05,64	8:25:45,05
8	0:00:00,08	0:00:01,49	0:00:40,60	0:17:44,19	8:28:24,56
9	0:00:00,08	0:00:01,47	0:00:42,89	0:21:01,89	8:20:30,74
10	0:00:00,09	0:00:01,52	0:00:42,69	0:17:48,85	8:48:17,73
11	0:00:00,04	0:00:01,62	0:00:37,33	0:20:59,36	8:38:18,39
12	0:00:00,06	0:00:01,44	0:00:42,78	0:21:05,11	8:38:24,37
13	0:00:00,10	0:00:01,76	0:00:39,03	0:17:25,41	9:05:50,28
14	0:00:00,08	0:00:01,55	0:00:42,80	0:17:07,86	8:33:09,85
15	0:00:00,09	0:00:01,66	0:00:42,43	0:20:46,41	8:50:21,32
16	0:00:00,07	0:00:01,79	0:00:40,82	0:16:58,07	8:22:55,42
17	0:00:00,09	0:00:01,85	0:00:41,22	0:21:20,09	8:41:17,71
18	0:00:00,07	0:00:01,57	0:00:38,11	0:17:18,09	9:06:45,96
19	0:00:00,11	0:00:01,40	0:00:38,13	0:18:27,51	9:06:34,60
20	0:00:00,70	0:00:01,53	0:00:38,74	0:18:49,21	9:00:17,71

На рисунке 8 изображён точечный график анализа времени обработок матриц, решённых методом полного перебора.

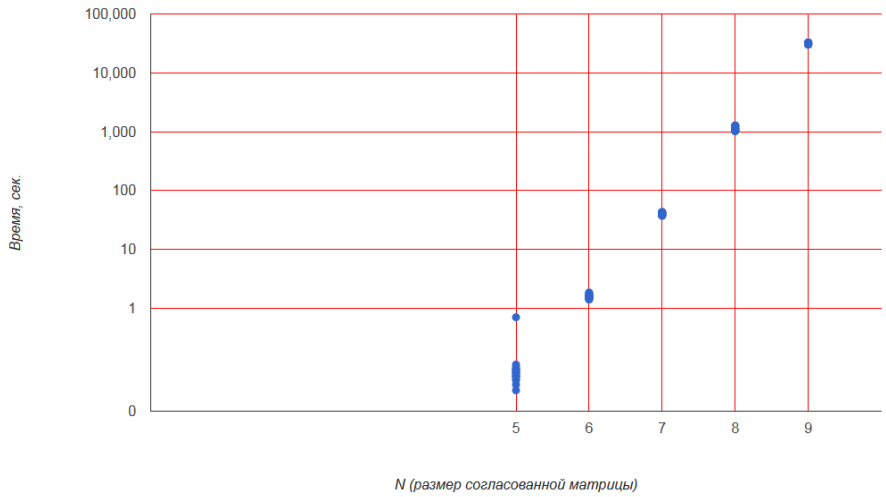

Рисунок 8 – Точечный логарифмический график анализа времён обработок матриц, решённых методом полного перебора

На рисунке 9 изображена гистограмма времен быстрых обработок среди 20 различных матриц, решённых методом полного перебора, а также их возможных задержек.

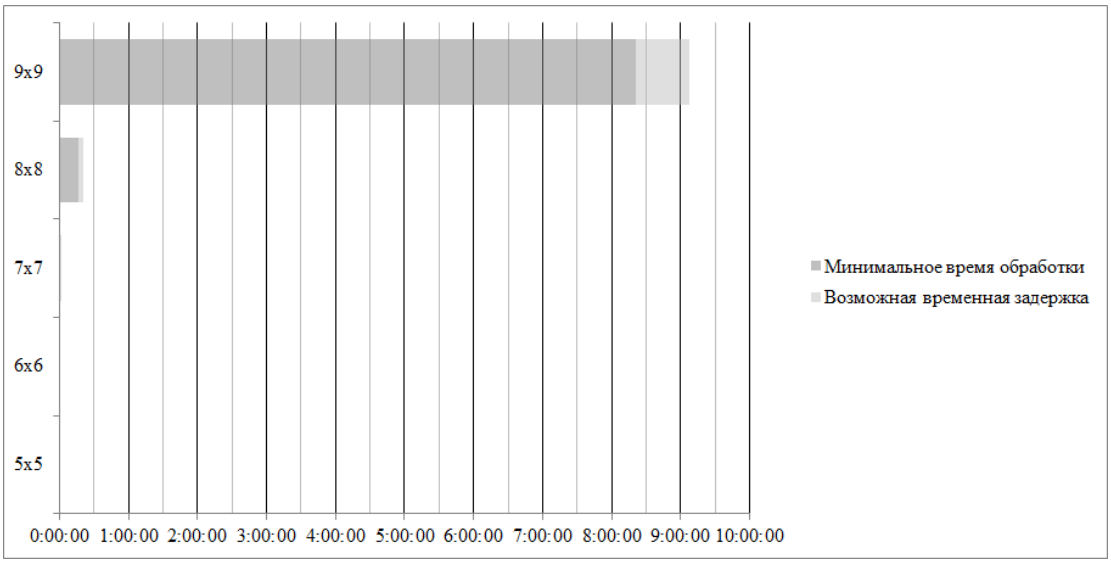

Рисунок 9 – Гистограмма времён быстрых обработок матриц, решённых методом полного перебора и их возможных задержек

На рисунке 10 изображена гистограмма общих времён обработок всех 20 матриц, решённых методом полного перебора.

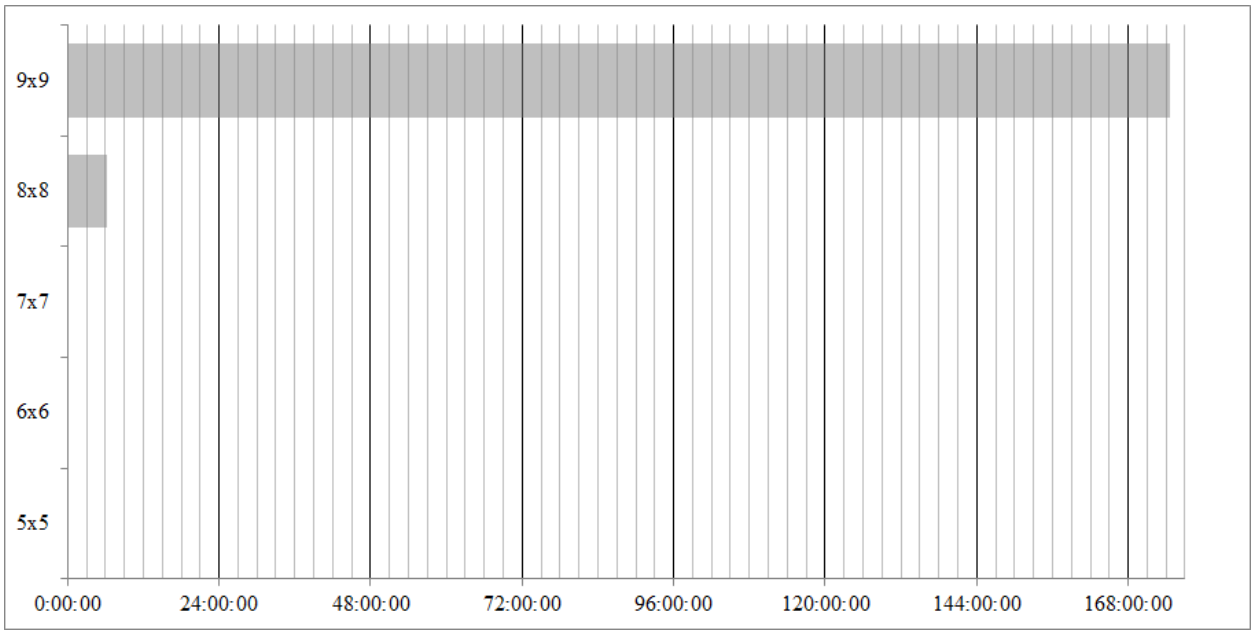

Рисунок 10 – Гистограмма общих времён обработок всех 20-ти матриц, решённых методом полного перебора

Вывод: метод полного перебора, в основном, помогает перебирать все варианты на случай, если претендентам предоставят выбор. Но он крайне неэффективен по причине того, что матрицы, чья размерность 9x9 или более, занимают более 8 часов на полную обработку. А в случае с 20-тью случайными матрицами и вовсе это затянулось более, чем на 7 суток.

3.3 Венгерский метод

На таблицах 4-5 изображены времена, за которые венгерским методом были решены 20 различных матриц, чья размерность – 250x250, 500x500, …, 4000x4000.

Таблица 4 – Результаты обработок 20 матриц венгерским методом (1-ая часть)

Кол-во прет-тов	250	500	750	1000	1250	1500	1750	2000
1	0:01,15	0:04,59	0:14,10	0:20,70	0:32,23	1:11,45	1:05,73	2:02,09
2	0:01,06	0:04,56	0:09,31	0:25,40	0:37,88	0:55,30	1:06,32	2:01,47
3	0:00,97	0:04,56	0:13,32	0:13,42	0:29,22	0:44,94	0:58,91	1:53,84
4	0:01,28	0:05,18	0:13,54	0:24,60	0:31,93	0:48,31	1:32,74	1:39,32
5	0:01,19	0:06,43	0:10,41	0:20,97	0:29,19	0:53,13	1:05,14	2:02,99
6	0:01,26	0:04,07	0:13,79	0:22,06	0:29,05	0:59,67	1:19,40	1:34,58
7	0:01,08	0:04,18	0:12,36	0:18,86	0:37,77	0:57,17	1:01,20	1:54,91
8	0:00,95	0:06,93	0:13,46	0:26,19	0:29,16	0:54,54	1:27,11	1:36,69
9	0:01,36	0:05,04	0:12,57	0:23,98	0:32,10	0:59,87	1:20,73	1:46,88
10	0:01,15	0:06,24	0:13,75	0:20,28	0:31,93	1:03,08	1:17,75	2:11,41
11	0:01,05	0:05,01	0:09,83	0:21,81	0:29,42	0:54,21	1:10,79	1:58,71
12	0:01,00	0:04,10	0:12,75	0:24,15	0:31,93	0:49,39	1:03,41	1:23,99
13	0:00,97	0:04,18	0:11,68	0:20,92	0:42,90	0:54,41	0:54,83	2:32,52
14	0:01,36	0:05,05	0:10,64	0:22,89	0:32,64	0:57,45	1:25,19	2:39,10
15	0:01,44	0:05,35	0:11,45	0:20,90	0:21,18	0:50,40	1:06,52	2:09,07
16	0:00,87	0:05,54	0:15,74	0:22,82	0:35,35	0:31,78	1:19,31	1:27,74
17	0:01,36	0:05,57	0:11,34	0:24,45	0:47,87	0:45,46	1:01,43	1:20,19
18	0:00,86	0:06,74	0:11,64	0:18,86	0:52,78	1:11,44	1:22,09	1:39,74
19	0:01,06	0:05,18	0:10,87	0:18,89	0:41,92	1:23,03	1:31,17	1:31,70
20	0:00,87	0:04,15	0:10,12	0:22,25	0:55,81	0:52,19	1:34,09	1:30,09

Таблица 5 – Результаты обработок 20 матриц венгерским методом (2-ая часть)

Кол-во прет-тов	2250	2500	2750	3000	3250	3500	3750	4000
1	2:30,11	2:11,56	2:42,50	3:23,91	3:46,18	4:14,08	5:57,97	6:12,40
2	2:31,42	2:05,05	3:12,73	2:37,91	3:26,28	5:34,97	6:03,94	6:04,19

Продолжение таблицы 5

Кол-во прет-тов	2250	2500	2750	3000	3250	3500	3750	4000
3	1:57,68	2:39,64	2:04,72	3:22,54	2:47,44	3:42,67	6:30,22	6:04,44
4	2:04,01	2:39,63	2:59,29	2:42,03	3:10,93	3:39,00	4:53,42	5:46,02
5	1:52,59	1:56,44	3:27,41	3:20,40	4:10,64	4:47,97	5:13,98	4:31,85
6	1:45,94	2:06,33	3:17,17	3:31,30	3:27,76	4:39,96	5:09,38	5:36,59
7	1:43,55	3:02,71	2:45,05	3:40,32	4:29,43	4:23,56	5:37,53	6:35,25
8	2:04,19	2:12,67	2:44,73	3:53,89	3:48,21	4:39,99	4:16,52	5:48,49
9	2:02,54	2:27,34	2:28,22	3:17,72	3:47,80	3:55,25	4:56,90	6:22,19
10	2:30,86	2:17,15	2:27,79	2:55,51	3:45,97	5:01,76	5:36,13	5:41,76
11	2:08,13	2:18,11	3:00,09	2:55,66	3:39,60	4:23,52	5:38,31	5:38,39
12	1:37,09	1:52,43	2:54,87	2:38,36	3:00,23	4:24,80	6:17,17	4:43,69
13	2:39,88	2:16,73	2:57,61	2:36,65	3:44,23	4:19,93	6:07,71	5:42,47
14	1:53,44	1:28,29	2:44,81	2:20,64	2:44,23	3:57,92	5:39,17	5:35,86
15	2:33,84	1:51,61	2:29,85	2:21,41	3:42,60	3:11,38	4:18,01	5:07,90
16	2:16,62	3:13,10	2:27,33	2:08,10	4:02,43	3:57,57	4:12,46	5:38,39
17	1:57,49	2:10,82	1:45,35	3:33,67	3:23,36	5:09,74	5:02,60	4:06,03
18	2:12,27	2:10,78	2:44,13	2:39,80	3:43,48	3:59,47	5:25,50	6:08,65
19	2:10,36	2:31,20	2:44,58	2:57,67	3:32,17	5:05,93	3:51,35	6:07,69
20	2:19,50	2:57,91	3:11,88	3:11,99	4:01,69	4:20,24	5:32,80	6:06,54

На рисунке 11 изображён точечный график анализа времён обработок матриц, решённых венгерским методом.

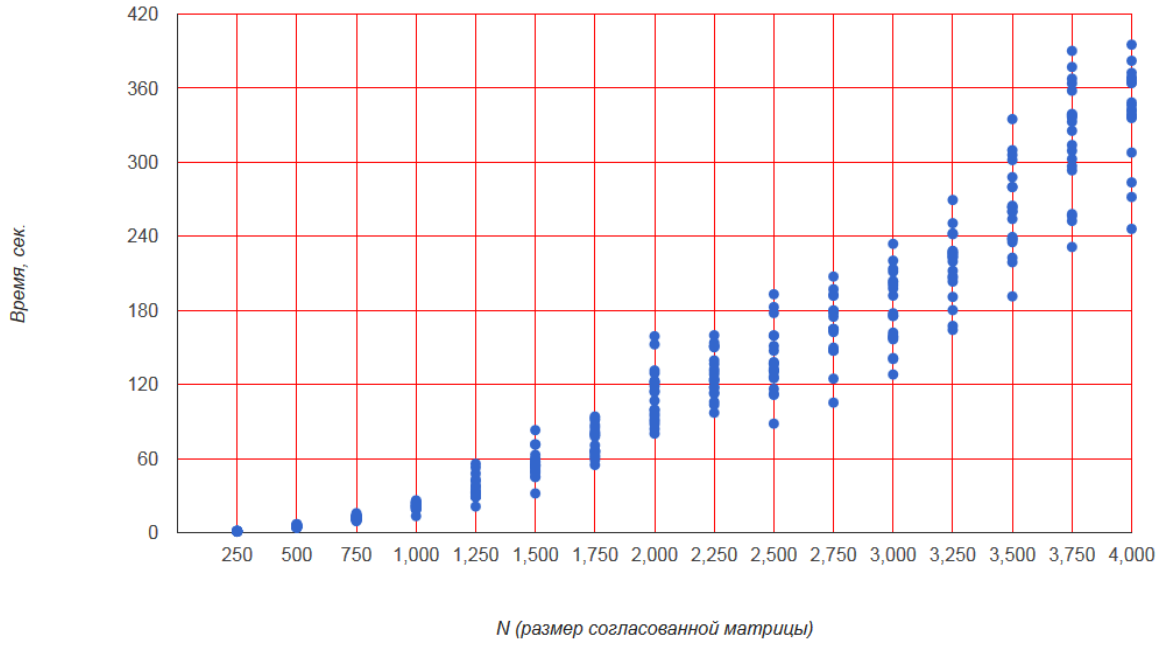

Рисунок 11 – Точечный график анализа времён обработок матриц, решённых венгерским методом

На рисунке 12 изображена гистограмма времен быстрых обработок среди 20 различных матриц, решённых венгерским методом, а также их возможных задержек.

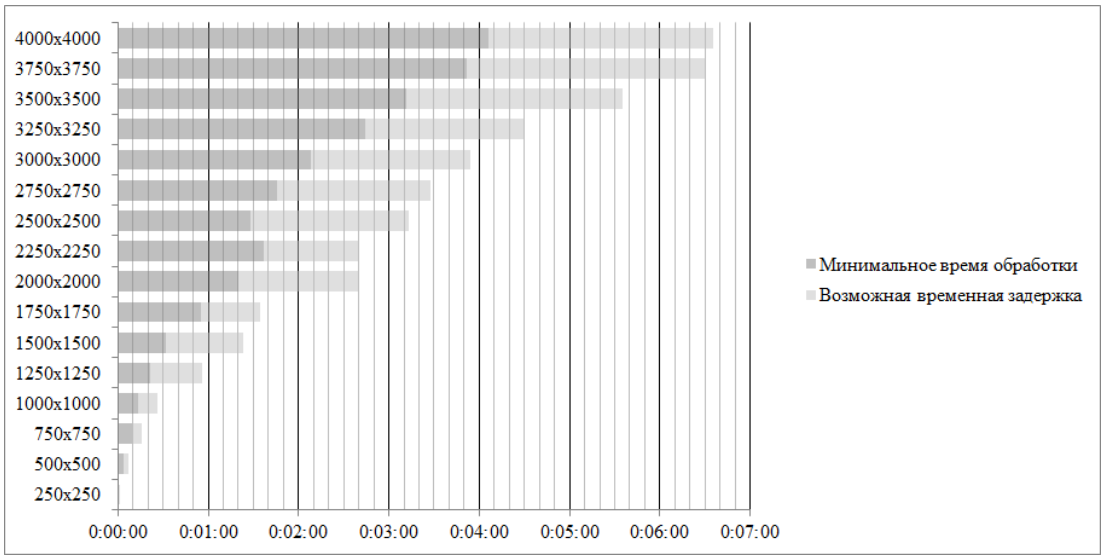

Рисунок 12 – Гистограмма времён быстрых обработок матриц, решённых венгерским методом, и их возможных задержек

На рисунке 13 изображена гистограмма общих времён обработок всех 20 матриц, решённых венгерским методом.

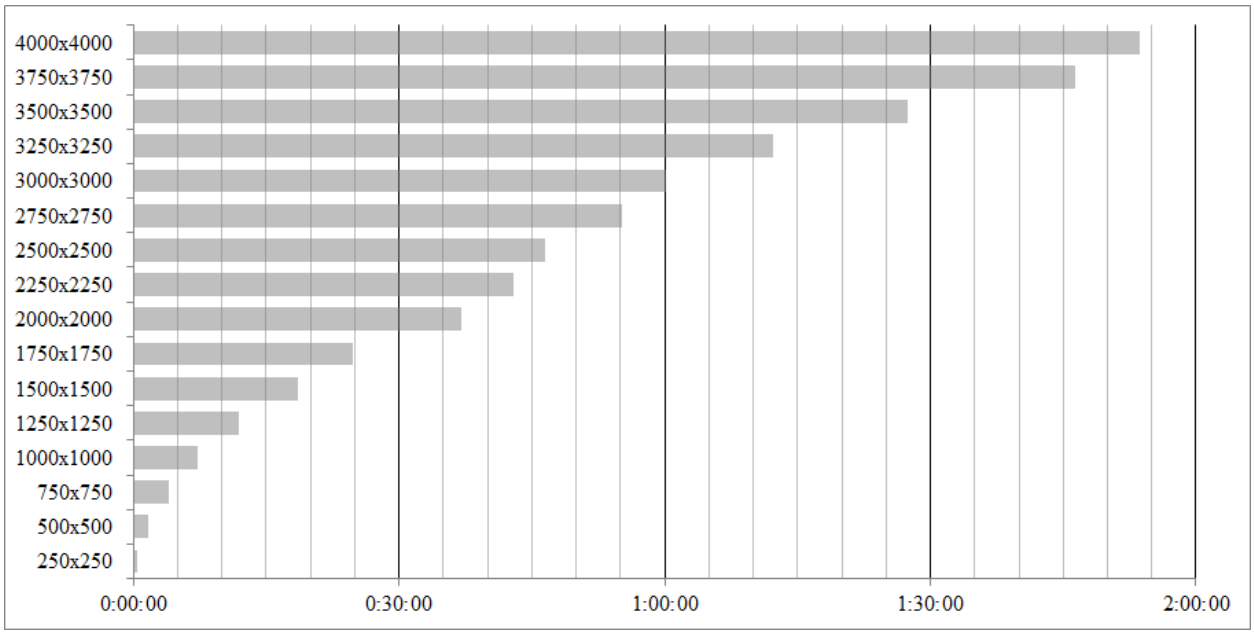

Рисунок 13 – Гистограмма общих времён обработок всех 20-ти матриц, решённых венгерским методом

Вывод: этот метод показывает наиболее быструю обработку даже при матрицах от площади в 1000x1000 до 4000x4000. Но, с другой стороны, он выводит только один оптимальный вариант, и это сильно разочаровывает.

3.4 Метод потенциалов

На таблице 6 изображены времена, за которые методом потенциалов транспортной задачи были решены 4 различных матрицы, чья размерность – 50x50, 100x100, 150x150, 200x200, при ограничениях $b_i = B_i = 1$ (i = 1,2,…, n).

Таблица 6 – Результаты обработок 20 матриц методом потенциалов транспортной задачи

Кол-во прет-тов	50x50	100x100	150x150	200x200
1	0:10,32	1:26,43	6:02,44	15:18,69
2	0:09,79	1:29,10	5:13,98	15:09,58
3	0:09,39	1:32,83	5:34,21	14:48,31
4	0:09,22	1:21,34	5:39,60	14:12,44
5	0:07,85	1:24,76	5:28,02	15:47,14
6	0:08,53	1:21,68	5:28,67	15:03,73
7	0:08,70	1:18,11	6:17,96	14:32,82
8	0:09,13	1:22,74	5:31,61	14:40,19
9	0:08,99	1:24,78	5:31,33	14:49,83
10	0:08,40	1:27,50	5:20,93	14:22,95
11	0:08,71	1:20,85	5:20,49	15:06,57
12	0:08,29	1:20,67	5:49,22	14:22,02
13	0:08,76	1:25,94	5:58,32	14:30,94
14	0:09,25	1:25,81	5:40,00	15:22,13
15	0:08,88	1:20,05	5:50,19	14:49,55
16	0:08,47	1:29,05	6:01,36	14:31,25
17	0:10,26	1:18,40	5:41,63	15:26,03
18	0:09,79	1:27,56	5:49,13	14:32,48
19	0:07,49	1:29,01	5:57,26	14:56,67
20	0:09,40	1:30,34	6:21,85	14:13,54

На рисунке 14 изображён точечный график анализа времён обработок матриц, решённых методом потенциалов транспортной задачи.

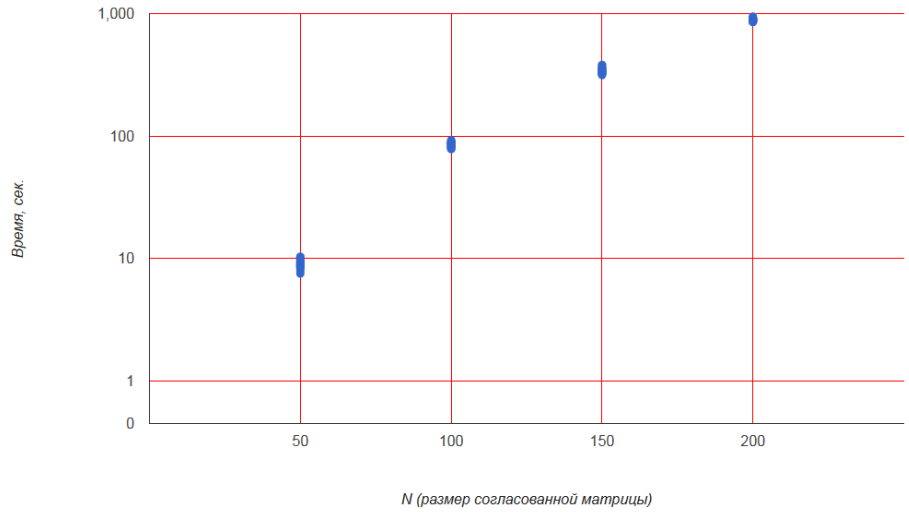

Рисунок 14 – Точечный логарифмический график анализа времён обработок матриц, решённых методом потенциалов транспортной задачи

На рисунке 15 изображена гистограмма времен быстрых обработок среди 20 различных матриц, решённых методом потенциалов транспортной задачи, а также их возможных задержек.

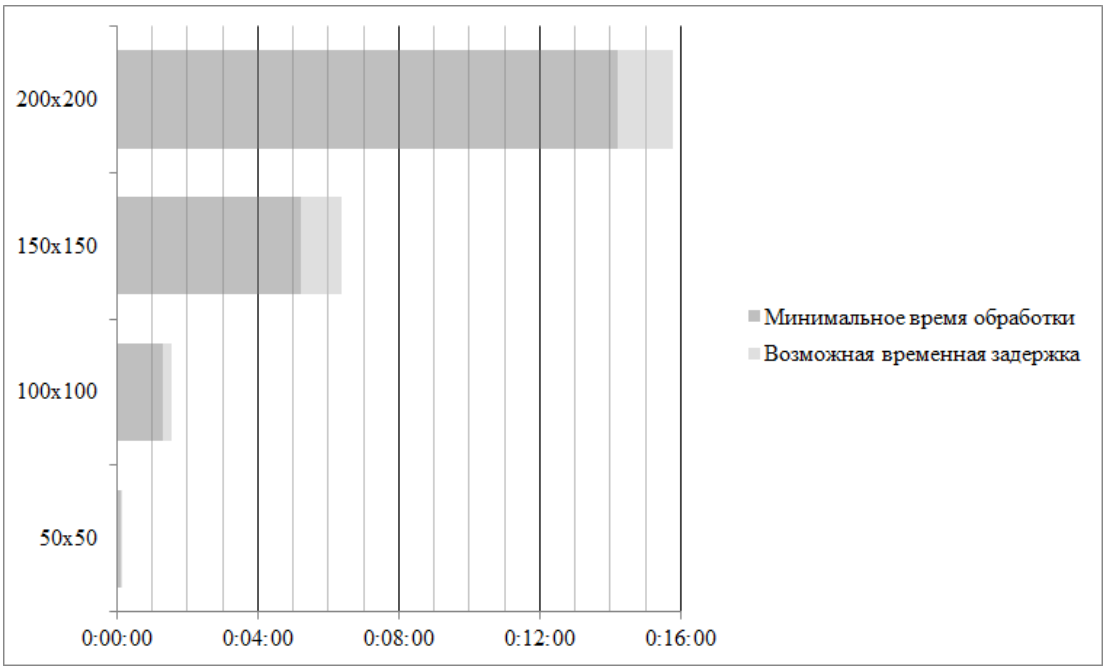

Рисунок 15 – Гистограмма времён быстрых обработок матриц, решённых методом потенциалов транспортной задачи, и их возможных задержек

На рисунке 16 изображена гистограмма общих времён обработок всех 20 матриц, решённых методом потенциалов транспортной задачи.

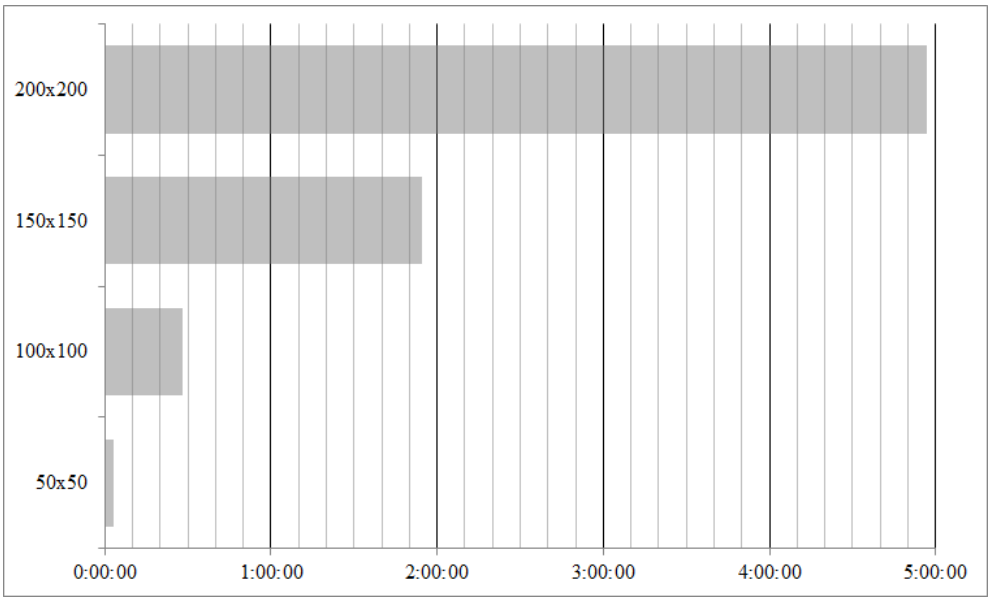

Рисунок 16 – Гистограмма общих времён обработок всех 20-ти матриц, решённых методом потенциалов транспортной задачи

На таблице 7 изображены времена, за которые методом потенциалов транспортной задачи были решены 4 различных матрицы, чья размерность – 400x2, 800x2, 1200x2, 1600x2, при ограничениях $b_1 = b_2 = B_1 = B_2 = n/2$.

Таблица 7 – Результаты обработок 20 матриц методом потенциалов транспортной задачи

Кол-во прет-тов	400x2	800x2	1200x2	1600x2
1	8,09	49,66	159,20	446,82
2	8,69	51,08	170,62	475,81
3	9,63	58,48	161,82	388,44
4	8,45	53,78	184,44	420,43
5	8,78	54,38	189,32	394,09

Продолжение таблицы 7

Кол-во прет-тов	400x2	800x2	1200x2	1600x2
6	9,20	50,14	168,77	348,97
7	7,70	51,37	167,23	345,61
8	9,77	49,55	184,98	377,59
9	7,33	56,23	203,81	422,16
10	7,27	58,03	209,58	344,68
11	8,19	52,47	189,54	340,74
12	8,43	55,01	187,78	348,86
13	8,22	55,95	183,58	379,84
14	8,39	53,40	163,73	338,77
15	9,40	52,71	166,60	401,56
16	9,20	51,82	167,42	374,31
17	8,65	48,61	201,88	409,60
18	7,99	54,41	206,36	327,11
19	7,36	51,06	221,31	371,52
20	8,35	88,29	209,68	387,93

На рисунке 17 изображён точечный график анализа времён обработок матриц, решённых методом потенциалов транспортной задачи.

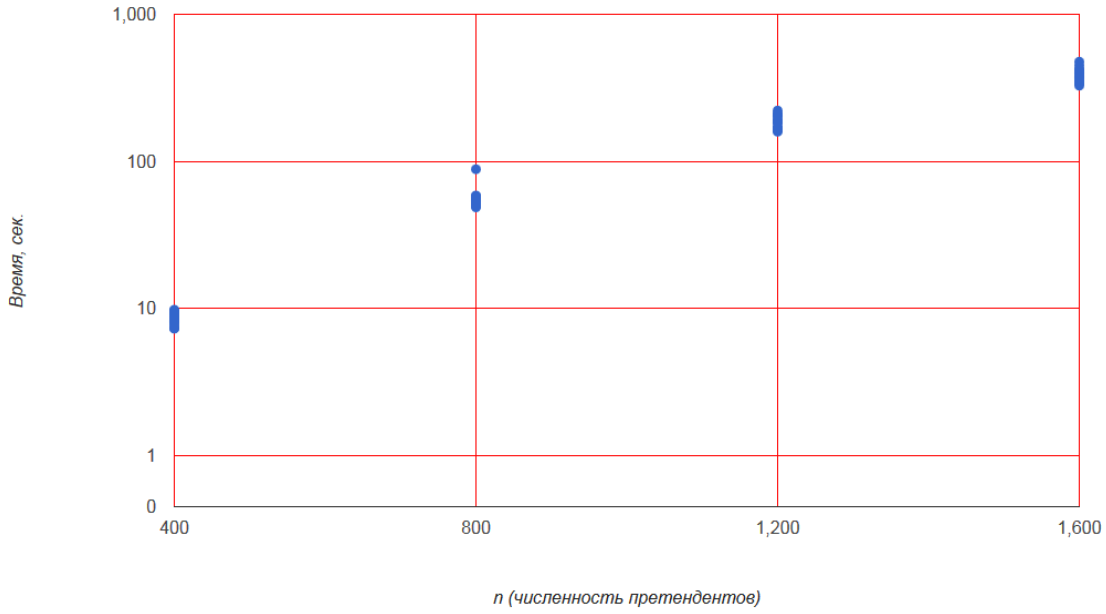

Рисунок 17 – Точечный логарифмический график анализа времён обработок матриц, решённых методом потенциалов транспортной задачи

На рисунке 18 изображена гистограмма времен быстрых обработок среди 20 различных матриц, решённых методом потенциалов транспортной задачи, а также их возможных задержек.

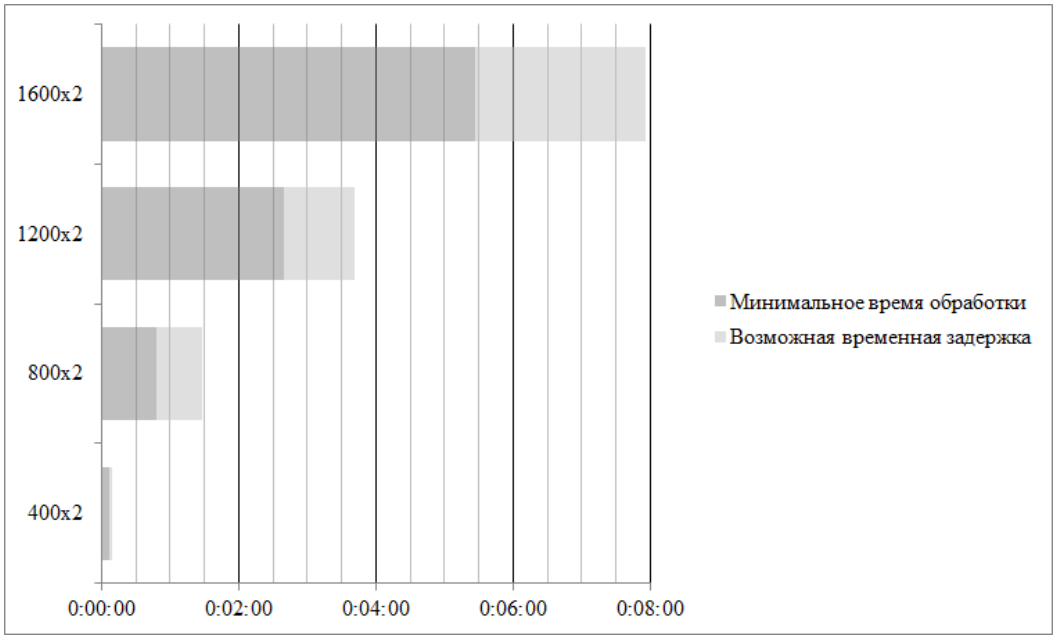

Рисунок 18 – Гистограмма времён быстрых обработок матриц, решённых методом потенциалов транспортной задачи, и их возможных задержек

На рисунке 19 изображена гистограмма общих времён обработок всех 20 матриц, решённых методом потенциалов транспортной задачи.

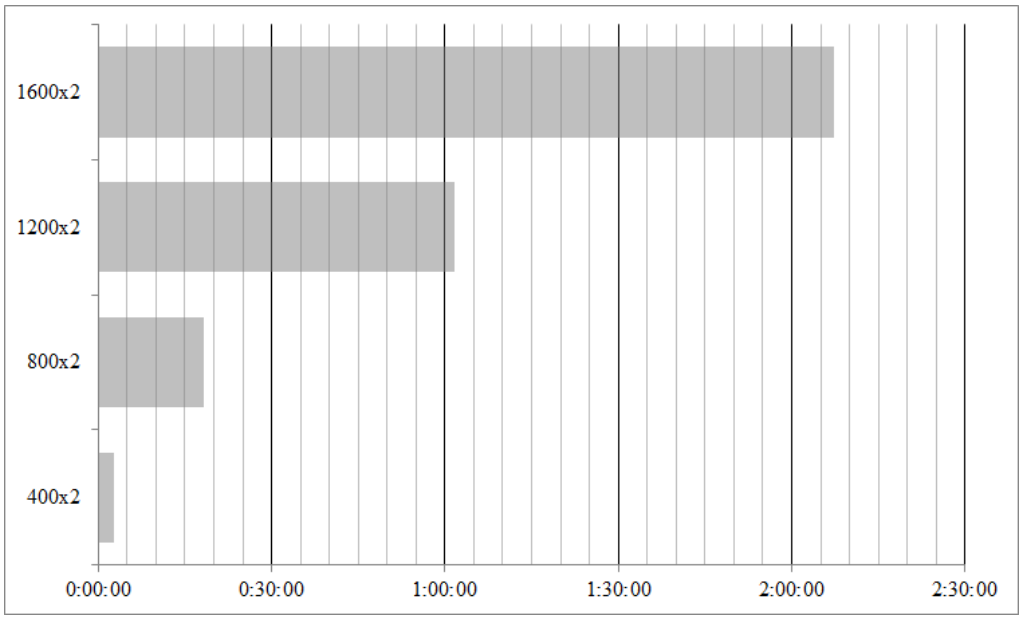

Рисунок 19 – Гистограмма общих времён обработок всех 20-ти матриц, решённых методом потенциалов транспортной задачи

Вывод: метод потенциалов более эффективный, чем метод полного перебора, но менее эффективный, чем венгерский метод. Да, он может обрабатывать быстрее матрицы большей размерности, но не более чем 200 претендентов. В наиболее оптимальных для метода потенциалов экспериментах (с 2-мя помещениями и ограничениями $b_1 = b_2 = B_1 = B_2 = n/2$) максимальная размерность задачи увеличивается до 1600. Тем не менее, время расчётов остаётся на порядок больше, чем у венгерского метода.

ЗАКЛЮЧЕНИЕ

Целью работы являлось разработка алгоритма решения задачи о распределении сотрудников по помещениям. На основе проанализированной литературных источников было принято решение использовать венгерский метод для поиска оптимального решения.

Формализация задачи о распределении сотрудников по помещениям привела к постановке задачи линейного целочисленного математического программирования. Разработан алгоритм, позволяющий свести постановку задачи к классической задачи о назначениях.

Было проведено тестирование на синтетических данных. В результате показано значительное превосходство разработанного алгоритма по сравнению полным перебором или методом потенциалов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Акулич, И. Л. Математическое программирование в примерах и задачах: учеб. пособие для студентов экономических специальностей вузов / И. Л. Акулич. – Москва : Высшая школа, 2011. – 352 стр.

2. Пападимитриу, Х. Комбинаторная оптимизация. Алгоритмы и сложность / Х. Пападимитриу, К. Стайглиц. – Москва : Мир, 1984. – 510 стр.

3. Триус, Е. Б. Задачи математического программирования транспортного типа / Е. Б. Триус. – Москва : Советское радио, 1967. – 208 стр.

4. Гольштейн, Е. Г. Задачи линейного программирования транспортного типа / Е. Г. Гольштейн, Д. Б. Юдин. – Москва : Наука, ФИЗМАТЛИТ, 1969. – 384 стр.

5. Кристофидес, Н. Теория графов. Алгоритмический подход / Н. Кристофидес. – Москва : Мир, 1978. – 432 стр.

6. Python 3 для начинающих [Электронный ресурс]. – Режим доступа: https://pythonworld.ru (дата обращения: 04.05.2020).

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Базовая кафедра «Интеллектуальные системы управления»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Ю.Ю. Якунин
подпись инициалы, фамилия
«22» июня 2020г.

БАКАЛАВРСКАЯ РАБОТА

27.03.03 «Системный анализ и управление»

Алгоритм решения расширенной задачи о назначениях

Руководитель _____ ,25.06.20 доцент, канд. техн. наук А.А. Даничев_
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 25.06.20 Д.А. Колосков
подпись, дата инициалы, фамилия

Красноярск 2020