**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc199844084)

[1 МЕТОД ПАРЕТО 4](#_Toc199844085)

[1.1 Выбор Парето-оптимального множества 4](#_Toc199844086)

[1.2 Указание верхних/нижних границ критериев. 5](#_Toc199844087)

[1.3 Субоптимизация 6](#_Toc199844088)

[1.4 Лексикографическая оптимизация 6](#_Toc199844089)

[1.5 Результаты работы программы 7](#_Toc199844090)

[1.6 Вывод по методу Парето 8](#_Toc199844091)

[2 МЕТОД ЭЛЕКТРА II 9](#_Toc199844092)

[2.1 Выбор лучшего варианта 9](#_Toc199844093)

[2.2 Веса предпочтений 10](#_Toc199844094)

[2.3 Результат работы программы 23](#_Toc199844095)

[2.4 Вывод по методу Электра II 24](#_Toc199844096)

[3 МЕТОД АНАЛИЗА ИЕРАРХИЙ 25](#_Toc199844097)

[3.1 Постановка задачи 25](#_Toc199844098)

[3.2 Представление проблемы в виде иерархии 25](#_Toc199844099)

[3.3 Установка приоритетов критериев 27](#_Toc199844100)

[3.4 Синтез приоритетов 27](#_Toc199844101)

[3.5 Согласованность локальных приоритетов 34](#_Toc199844102)

[3.6 Синтез альтернатив 39](#_Toc199844103)

[3.7 Результаты работы программы 40](#_Toc199844104)

[3.8 Вывод по методу МАИ 41](#_Toc199844105)

[4 ГРАФИЧЕСКИЙ МЕТОД 42](#_Toc199844106)

[4.1 Постановка задачи 42](#_Toc199844107)

[4.2 Данные индивидуального варианта 42](#_Toc199844108)

[4.3 Подготовка данных 42](#_Toc199844109)

[4.4 Построение графика 44](#_Toc199844110)

[4.5 Выделение области допустимых решений 45](#_Toc199844111)

[4.6 Максимум функции 46](#_Toc199844112)

[4.7 Минимум функции 48](#_Toc199844113)

[4.8 Вывод по графическому методу 50](#_Toc199844114)

[5 СИМПЛЕКСНЫЙ МЕТОД 51](#_Toc199844115)

[5.1 Постановка задачи 51](#_Toc199844116)

[5.2 Математическая модель задачи 51](#_Toc199844117)

[5.3 Ручной расчет метода 51](#_Toc199844118)

[5.4 Консольный результат работы 57](#_Toc199844119)

[5.5 Вывод по симплексному методу 57](#_Toc199844120)

[6 ДВОЙСТВЕННАЯ ЗАДАЧА 58](#_Toc199844121)

[6.1 Постановка задачи 58](#_Toc199844122)

[6.2 Математическая модель исходной задачи 59](#_Toc199844123)

[6.3 Соответствующая исходной двойственная задача 59](#_Toc199844124)

[6.4 Первая теорема двойственности 60](#_Toc199844125)

[6.5 Вторая теорема двойственности 62](#_Toc199844126)

[6.6 Третья теорема двойственности 64](#_Toc199844127)

[6.7 Консольный результат программы 69](#_Toc199844128)

[6.8 Вывод по двойственному методу 69](#_Toc199844129)

[ЗАКЛЮЧЕНИЕ 70](#_Toc199844130)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 71](#_Toc199844131)

[ПРИЛОЖЕНИЯ 72](#_Toc199844132)

ВВЕДЕНИЕ

Курсовая работа посвящена применению методов многокритериальной оптимизации и линейного программирования для решения практических задач выбора оптимальных решений. В работе рассматриваются такие методы, как Парето, Электра II, анализ иерархий, симплекс-метод, двойственная задача и транспортная задача, которые применяются для анализа различных альтернатив на основе заданных критериев. Основной целью исследования является выбор оптимального варианта дизельного топлива от различных поставщиков с учетом таких параметров, как цетановое число, цетановый индекс, температура вспышки и кинематическая вязкость, а также решение задачи оптимального планирования производства и распределения ресурсов. Работа демонстрирует, как эти методы позволяют систематизировать процесс принятия решений, минимизировать затраты и учитывать компромиссы между критериями. Анализ включает как теоретическое обоснование методов, так и их практическую реализацию, подкрепленную программными расчетами.

1. МЕТОД ПАРЕТО

Метод Парето представляет собой сравнение критериев различных альтернатив. При сравнении берутся все критерии и сравниваются образом доминации.

Если положительный критерий у альтернативы А больше, чем у альтернативы Б, то альтернатива А считается доминирующей, в противном случае доминированной.

Если отрицательный критерий у альтернативы А больше, чем у альтернативы Б, то альтернатива Б считается доминирующей, иначе считается доминированной.

Если не все критерии у альтернативы А считаются доминирующими, то такая альтернатива считается несравнимой с альтернативой Б.

## **Выбор Парето-оптимального множества**

Приведём пример выбора молока в магазине с использованием Парето оптимального множества решений. (Таблица 1)

*Таблица 1.1.1 – Пример таблицы*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Альтернативы | Цена (руб)(-) | Вес (г)(+) | Калорийность(-) | Срок хранения(+) |
| 1.Простоквашино | 99 | 930 | 53 | 16 |
| 2.Домик в деревне | 89 | 930 | 53 | 15 |
| 3.Молочный знак | 71 | 900 | 53 | 14 |
| 4.ЭкоНива | 114 | 1000 | 60 | 365 |
| 5.Просто | 85 | 970 | 53 | 180 |
| 6.Сарафаново | 129 | 970 | 46 | 180 |
| 7.Зелёная линия | 79 | 900 | 40 | 10 |
| 8.Искренне ваш | 85 | 930 | 53 | 16 |
| 9.Экомилк | 89 | 900 | 70 | 21 |

Сравним попарно все альтернативы и сведём их в таблицу 2

*Таблица 1.1.2 – Сравнение*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | x | x | x | x | x | x | x | x | x |
| 2 | н | x | x | x | x | x | x | x | x |
| 3 | н | н | x | x | x | x | x | x | x |
| 4 | н | н | н | x | x | x | x | x | x |
| 5 | A5 | A5 | н | н | x | x | x | x | x |
| 6 | н | н | н | н | н | x | x | x | x |
| 7 | н | н | н | н | н | н | x | x | x |
| 8 | А8 | A8 | н | н | н | н | н | x | x |
| 9 | н | н | н | н | н | н | н | н | x |

# Указание верхних/нижних границ критериев.

Установим для таблицы нижнюю границу: вес не менее 950 и срок хранения не менее 20

*Таблица 1.2.1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Альтернативы | Цена (руб)(-) | Вес (г)(+) | Калорийность(-) | Срок хранения(+) |
| 4.ЭкоНива | 114 | 1000 | 60 | 365 |
| 5.Просто | 85 | 970 | 53 | 180 |
| 6.Сарафаново | 129 | 970 | 46 | 180 |

Из них оптимальными по Парето является 5.

**1.3 Субоптимизация**

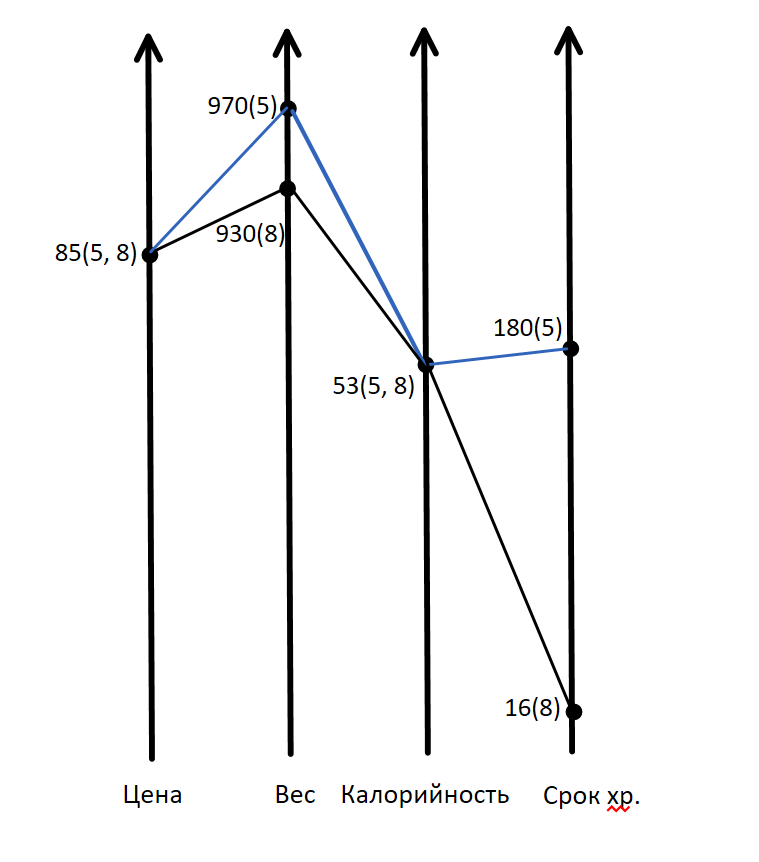
Пусть в качестве главного критерия выступает критерий цена   
Вес не менее 930, каллорийность не менее 53, срок хранения не менее 20

*Таблица 1.3.1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Альтернативы | Цена (руб)(-) | Вес (г)(+) | Калорийность(-) | Срок хранения(+) |
| 4.ЭкоНива | 114 | 1000 | 60 | 365 |
| 5.Просто | 85 | 970 | 53 | 180 |

Из таблицы 4 видно что остались варианты 4 и 5 из них вариант 5 имеет меньшую цену

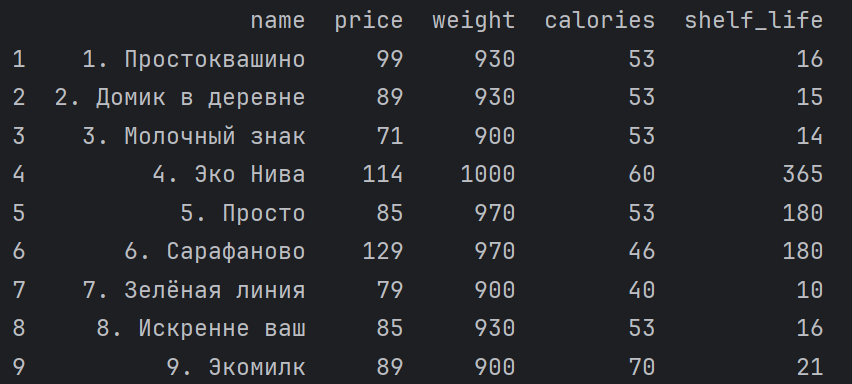
**1.4 Лексикографическая оптимизация**



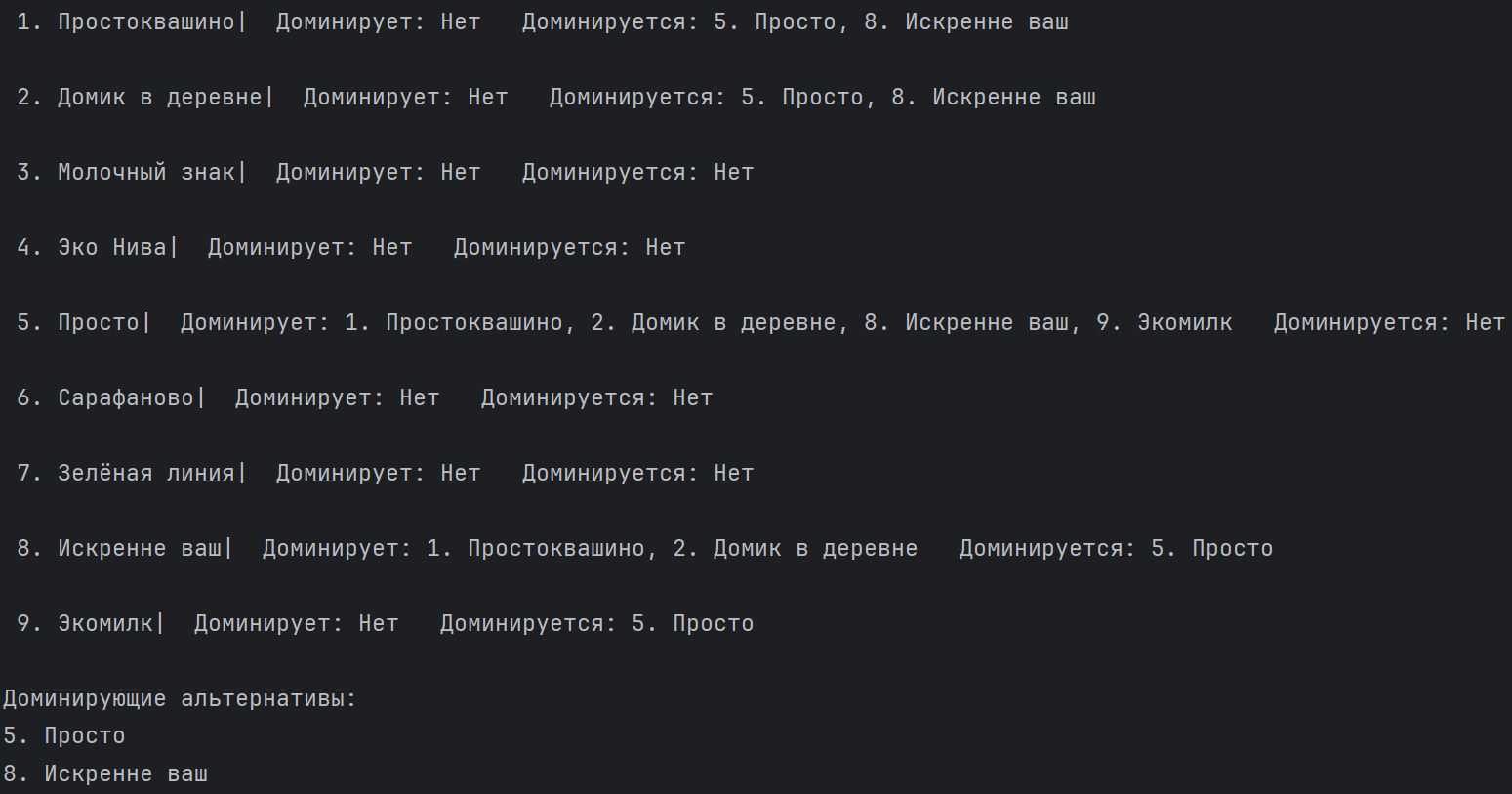
*Рисунок 1.4.1 - Лексикографическая оптимизация*

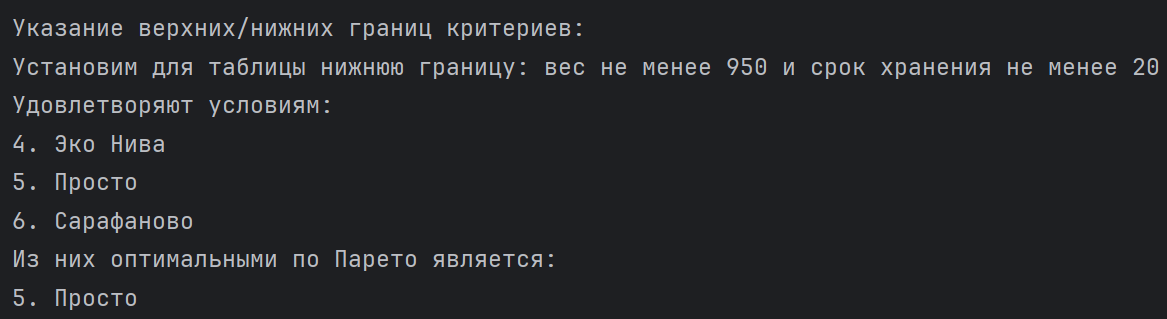
Исходя из графика лучший вариант 5

**1.5 Результаты работы программы**

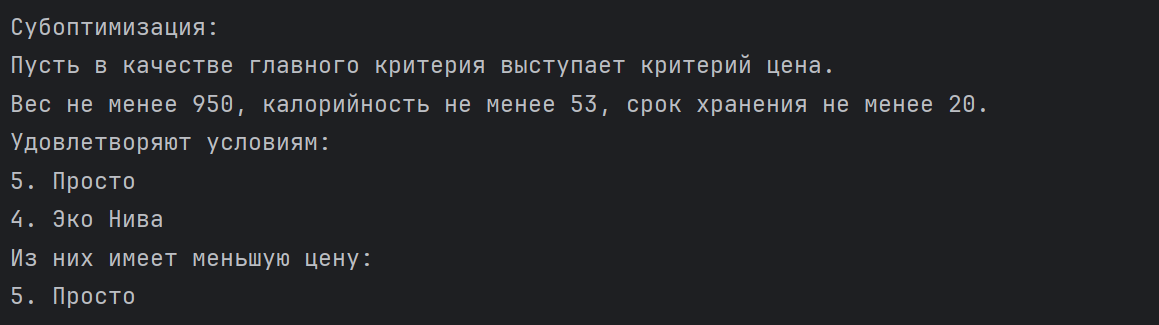
****

**Рисунок 1.5.1 –** Таблица альтернатив

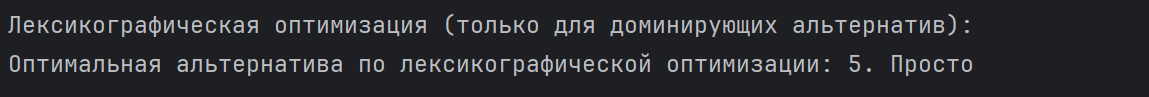
**Рисунок 1.5.2 –** Выбор Парето-оптимального множества



**Рисунок 1.5.3 –** Указание верхних/нижних границ критериев



**Рисунок 1.5.4 -** Субоптимизация



**Рисунок 1.5.5 -** Лексикографическая оптимизация

**1.6 Вывод по методу Парето**

В ходе выполнения данной работы рассмотрен метод Парето для многокритериального выбора оптимального варианта. В качестве примера использовалась задача выбора молока в магазине, для чего были выделены альтернативы, проанализированы их критерии.

Метод Парето является эффективным инструментом для предварительного отбора решений в многокритериальных задачах, но для окончательного выбора оптимального варианта зачастую необходимо применение дополнительных методов сужения множества решений, учитывающих предпочтения лица, принимающего решение (ЛПР).

# 2 МЕТОД ЭЛЕКТРА II

Метод Электра относится к группе методов многокритериального анализа решений и используется для выбора наилучшей альтернативы при наличии множества критериев. В данной работе применяется модификация метода — Электра II, которая учитывает как согласие, так и несогласие между альтернативами.

Метод Электра применяется в задачах, где необходимо учитывать противоречивые критерии и проводить ранжирование альтернатив с учетом пороговых значений предпочтений. Он широко используется в экономике, управлении проектами, логистике, экологическом мониторинге и других областях, где важно сравнивать альтернативы по нескольким параметрам.

В данной работе рассматриваются несколько альтернативных вариантов топлива с различными характеристиками. С помощью метода Электра II проводится их сравнительная оценка, позволяющая определить наиболее предпочтительные варианты.

## **2.1 Выбор лучшего варианта**

Составлена таблица критериев оценки проектов (Таблица 1).

*Таблица 2.1.1 – Таблица критериев для оценки альтернатив*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии | Вес критерия | Шкала | Код | Стремление |
| Цена | 5 | От 100 От 90 до 100  До 90 | 15  10  5 | min |
| Вес | 5 | От 1000 От 901 до 1000  До 901 | 15  10  5 | max |
| Калорийность | 4 | Высокая Средняя  Низкая | 15  10  5 | min |
| Срок хранения | 3 | Большой Средний  Маленький | 15  10  5 | max |

Составлена таблица оценок выбора лучшего молока в магазине. Для 9-ти альтернатив заполнена Таблицу 2.1.2.

*Таблица 2.1.2 – Таблица оценок по критериям*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Варианты решений | Критерии | | | |
| Цена | Вес | Калорийность | Срок хранения |
| 1 | Простоквашино | 10 | 10 | 10 | 10 |
| 2 | Домик в деревне | 5 | 10 | 10 | 10 |
| 3 | Молочный знак | 5 | 10 | 10 | 5 |
| 4 | ЭкоНива | 15 | 15 | 15 | 15 |
| 5 | Просто | 10 | 10 | 10 | 15 |
| 6 | Сарафаново | 15 | 10 | 5 | 15 |
| 7 | Зелёная линия | 5 | 5 | 5 | 10 |
| 8 | Искренне ваш | 10 | 10 | 10 | 10 |
| 9 | Экомилк | 10 | 5 | 15 | 15 |
| Вес | | 5 | 5 | 4 | 3 |
| Стремление | | min | max | min | max |

**2.2 Веса предпочтений**

Рассмотрим альтернативы 1 и 2 (i = 1, j = 2):

P12 = 0 + 0 + 0 + 0 = 0

N12 = 5 + 0 + 0 + 0 = 5

D12 = P12/N12 = 0/5 = 0.0 - inf - принимаем

P21 = 5 + 0 + 0 + 0 = 5

N21 = 0 + 0 + 0 + 0 = 0

D21 = N21/P21 = 5/0 - Деление на ноль, отбрасываем

Рассмотрим альтернативы 1 и 3 (i = 1, j = 3):

P13 = 0 + 0 + 0 + 3 = 3

N13 = 5 + 0 + 0 + 0 = 5

D13 = P13/N13 = 3/5 = 0.6 < 1 - отбрасываем

P31 = 5 + 0 + 0 + 0 = 5

N31 = 0 + 0 + 0 + 3 = 3

D31 = N31/31 = 5/3 = 1.7 > 1 - принимаем

Рассмотрим альтернативы 1 и 4 (i = 1, j = 4):

P14 = 5 + 0 + 4 + 0 = 9

N14 = 0 + 5 + 0 + 3 = 8

D14 = P14/N14 = 9/8 = 1.1 > 1 - принимаем

P41 = 0 + 5 + 0 + 3 = 8

N41 = 5 + 0 + 4 + 0 = 9

D41 = N41/P41 = 8/9 = 0.9 < 1 - отбрасываем

Рассмотрим альтернативы 1 и 5 (i = 1, j = 5):

P15 = 0 + 0 + 0 + 0 = 0

N15 = 0 + 0 + 0 + 3 = 3

D15 = P15/N15 = 0/3 = 0.0 - inf - принимаем

P51 = 0 + 0 + 0 + 3 = 3

N51 = 0 + 0 + 0 + 0 = 0

D51 = N51/P51 = 3/0 - Деление на ноль, отбрасываем

Рассмотрим альтернативы 1 и 6 (i = 1, j = 6):

P16 = 5 + 0 + 0 + 0 = 5

N16 = 0 + 0 + 4 + 3 = 7

D16 = P16/N16 = 5/7 = 0.7 < 1 - отбрасываем

P61 = 0 + 0 + 4 + 3 = 7

N61 = 5 + 0 + 0 + 0 = 5

D61 = N61/61 = 7/5 = 1.4 > 1 - принимаем

Рассмотрим альтернативы 1 и 7 (i = 1, j = 7):

P17 = 0 + 5 + 0 + 0 = 5

N17 = 5 + 0 + 4 + 0 = 9

D17 = P17/N17 = 5/9 = 0.6 < 1 - отбрасываем

P71 = 5 + 0 + 4 + 0 = 9

N71 = 0 + 5 + 0 + 0 = 5

D71 = N71/71 = 9/5 = 1.8 > 1 - принимаем

Рассмотрим альтернативы 1 и 8 (i = 1, j = 8):

D18 P и N равны - ливаем

P18 = 0 + 0 + 0 + 0 = 0

N18 = 0 + 0 + 0 + 0 = 0

D18 = P18/N18 = 0/0 - Деление на ноль, отбрасываем

P81 = 0 + 0 + 0 + 0 = 0

N81 = 0 + 0 + 0 + 0 = 0

D81 = N81/P81 = 0/0 - Деление на ноль, отбрасываем

Рассмотрим альтернативы 1 и 9 (i = 1, j = 9):

P19 = 0 + 5 + 4 + 0 = 9

N19 = 0 + 0 + 0 + 3 = 3

D19 = P19/N19 = 9/3 = 3.0 > 1 - принимаем

P91 = 0 + 0 + 0 + 3 = 3

N91 = 0 + 5 + 4 + 0 = 9

D91 = N91/P91 = 3/9 = 0.3 < 1 - отбрасываем

Рассмотрим альтернативы 2 и 3 (i = 2, j = 3):

P23 = 0 + 0 + 0 + 3 = 3

N23 = 0 + 0 + 0 + 0 = 0

D23 = P23/N23 = 3/0 - Деление на ноль, отбрасываем

P32 = 0 + 0 + 0 + 0 = 0

N32 = 0 + 0 + 0 + 3 = 3

D32 = N32/P32 = 0/3 - inf - принимаем

Рассмотрим альтернативы 2 и 4 (i = 2, j = 4):

P24 = 5 + 0 + 4 + 0 = 9

N24 = 0 + 5 + 0 + 3 = 8

D24 = P24/N24 = 9/8 = 1.1 > 1 - принимаем

P42 = 0 + 5 + 0 + 3 = 8

N42 = 5 + 0 + 4 + 0 = 9

D42 = N42/P42 = 8/9 = 0.9 < 1 - отбрасываем

Рассмотрим альтернативы 2 и 5 (i = 2, j = 5):

P25 = 5 + 0 + 0 + 0 = 5

N25 = 0 + 0 + 0 + 3 = 3

D25 = P25/N25 = 5/3 = 1.7 > 1 - принимаем

P52 = 0 + 0 + 0 + 3 = 3

N52 = 5 + 0 + 0 + 0 = 5

D52 = N52/P52 = 3/5 = 0.6 < 1 - отбрасываем

Рассмотрим альтернативы 2 и 6 (i = 2, j = 6):

P26 = 5 + 0 + 0 + 0 = 5

N26 = 0 + 0 + 4 + 3 = 7

D26 = P26/N26 = 5/7 = 0.7 < 1 - отбрасываем

P62 = 0 + 0 + 4 + 3 = 7

N62 = 5 + 0 + 0 + 0 = 5

D62 = N62/62 = 7/5 = 1.4 > 1 - принимаем

Рассмотрим альтернативы 2 и 7 (i = 2, j = 7):

P27 = 0 + 5 + 0 + 0 = 5

N27 = 0 + 0 + 4 + 0 = 4

D27 = P27/N27 = 5/4 = 1.2 > 1 - принимаем

P72 = 0 + 0 + 4 + 0 = 4

N72 = 0 + 5 + 0 + 0 = 5

D72 = N72/P72 = 4/5 = 0.8 < 1 - отбрасываем

Рассмотрим альтернативы 2 и 8 (i = 2, j = 8):

P28 = 5 + 0 + 0 + 0 = 5

N28 = 0 + 0 + 0 + 0 = 0

D28 = P28/N28 = 5/0 - Деление на ноль, отбрасываем

P82 = 0 + 0 + 0 + 0 = 0

N82 = 5 + 0 + 0 + 0 = 5

D82 = N82/P82 = 0/5 - inf - принимаем

Рассмотрим альтернативы 2 и 9 (i = 2, j = 9):

P29 = 5 + 5 + 4 + 0 = 14

N29 = 0 + 0 + 0 + 3 = 3

D29 = P29/N29 = 14/3 = 4.7 > 1 - принимаем

P92 = 0 + 0 + 0 + 3 = 3

N92 = 5 + 5 + 4 + 0 = 14

D92 = N92/P92 = 3/14 = 0.2 < 1 - отбрасываем

Рассмотрим альтернативы 3 и 4 (i = 3, j = 4):

P34 = 5 + 0 + 4 + 0 = 9

N34 = 0 + 5 + 0 + 3 = 8

D34 = P34/N34 = 9/8 = 1.1 > 1 - принимаем

P43 = 0 + 5 + 0 + 3 = 8

N43 = 5 + 0 + 4 + 0 = 9

D43 = N43/P43 = 8/9 = 0.9 < 1 - отбрасываем

Рассмотрим альтернативы 3 и 5 (i = 3, j = 5):

P35 = 5 + 0 + 0 + 0 = 5

N35 = 0 + 0 + 0 + 3 = 3

D35 = P35/N35 = 5/3 = 1.7 > 1 - принимаем

P53 = 0 + 0 + 0 + 3 = 3

N53 = 5 + 0 + 0 + 0 = 5

D53 = N53/P53 = 3/5 = 0.6 < 1 - отбрасываем

Рассмотрим альтернативы 3 и 6 (i = 3, j = 6):

P36 = 5 + 0 + 0 + 0 = 5

N36 = 0 + 0 + 4 + 3 = 7

D36 = P36/N36 = 5/7 = 0.7 < 1 - отбрасываем

P63 = 0 + 0 + 4 + 3 = 7

N63 = 5 + 0 + 0 + 0 = 5

D63 = N63/63 = 7/5 = 1.4 > 1 - принимаем

Рассмотрим альтернативы 3 и 7 (i = 3, j = 7):

P37 = 0 + 5 + 0 + 0 = 5

N37 = 0 + 0 + 4 + 3 = 7

D37 = P37/N37 = 5/7 = 0.7 < 1 - отбрасываем

P73 = 0 + 0 + 4 + 3 = 7

N73 = 0 + 5 + 0 + 0 = 5

D73 = N73/73 = 7/5 = 1.4 > 1 - принимаем

Рассмотрим альтернативы 3 и 8 (i = 3, j = 8):

P38 = 5 + 0 + 0 + 0 = 5

N38 = 0 + 0 + 0 + 3 = 3

D38 = P38/N38 = 5/3 = 1.7 > 1 - принимаем

P83 = 0 + 0 + 0 + 3 = 3

N83 = 5 + 0 + 0 + 0 = 5

D83 = N83/P83 = 3/5 = 0.6 < 1 - отбрасываем

Рассмотрим альтернативы 3 и 9 (i = 3, j = 9):

P39 = 5 + 5 + 4 + 0 = 14

N39 = 0 + 0 + 0 + 3 = 3

D39 = P39/N39 = 14/3 = 4.7 > 1 - принимаем

P93 = 0 + 0 + 0 + 3 = 3

N93 = 5 + 5 + 4 + 0 = 14

D93 = N93/P93 = 3/14 = 0.2 < 1 - отбрасываем

Рассмотрим альтернативы 4 и 5 (i = 4, j = 5):

P45 = 0 + 5 + 0 + 0 = 5

N45 = 5 + 0 + 4 + 0 = 9

D45 = P45/N45 = 5/9 = 0.6 < 1 - отбрасываем

P54 = 5 + 0 + 4 + 0 = 9

N54 = 0 + 5 + 0 + 0 = 5

D54 = N54/54 = 9/5 = 1.8 > 1 - принимаем

Рассмотрим альтернативы 4 и 6 (i = 4, j = 6):

P46 = 0 + 5 + 0 + 0 = 5

N46 = 0 + 0 + 4 + 0 = 4

D46 = P46/N46 = 5/4 = 1.2 > 1 - принимаем

P64 = 0 + 0 + 4 + 0 = 4

N64 = 0 + 5 + 0 + 0 = 5

D64 = N64/P64 = 4/5 = 0.8 < 1 - отбрасываем

Рассмотрим альтернативы 4 и 7 (i = 4, j = 7):

P47 = 0 + 5 + 0 + 3 = 8

N47 = 5 + 0 + 4 + 0 = 9

D47 = P47/N47 = 8/9 = 0.9 < 1 - отбрасываем

P74 = 5 + 0 + 4 + 0 = 9

N74 = 0 + 5 + 0 + 3 = 8

D74 = N74/74 = 9/8 = 1.1 > 1 - принимаем

Рассмотрим альтернативы 4 и 8 (i = 4, j = 8):

P48 = 0 + 5 + 0 + 3 = 8

N48 = 5 + 0 + 4 + 0 = 9

D48 = P48/N48 = 8/9 = 0.9 < 1 - отбрасываем

P84 = 5 + 0 + 4 + 0 = 9

N84 = 0 + 5 + 0 + 3 = 8

D84 = N84/84 = 9/8 = 1.1 > 1 - принимаем

Рассмотрим альтернативы 4 и 9 (i = 4, j = 9):

D49 P и N равны - ливаем

P49 = 0 + 5 + 0 + 0 = 5

N49 = 5 + 0 + 0 + 0 = 5

P94 = 5 + 0 + 0 + 0 = 5

N94 = 0 + 5 + 0 + 0 = 5

Рассмотрим альтернативы 5 и 6 (i = 5, j = 6):

P56 = 5 + 0 + 0 + 0 = 5

N56 = 0 + 0 + 4 + 0 = 4

D56 = P56/N56 = 5/4 = 1.2 > 1 - принимаем

P65 = 0 + 0 + 4 + 0 = 4

N65 = 5 + 0 + 0 + 0 = 5

D65 = N65/P65 = 4/5 = 0.8 < 1 - отбрасываем

Рассмотрим альтернативы 5 и 7 (i = 5, j = 7):

P57 = 0 + 5 + 0 + 3 = 8

N57 = 5 + 0 + 4 + 0 = 9

D57 = P57/N57 = 8/9 = 0.9 < 1 - отбрасываем

P75 = 5 + 0 + 4 + 0 = 9

N75 = 0 + 5 + 0 + 3 = 8

D75 = N75/75 = 9/8 = 1.1 > 1 - принимаем

Рассмотрим альтернативы 5 и 8 (i = 5, j = 8):

P58 = 0 + 0 + 0 + 3 = 3

N58 = 0 + 0 + 0 + 0 = 0

D58 = P58/N58 = 3/0 - Деление на ноль, отбрасываем

P85 = 0 + 0 + 0 + 0 = 0

N85 = 0 + 0 + 0 + 3 = 3

D85 = N85/P85 = 0/3 - inf - принимаем

Рассмотрим альтернативы 5 и 9 (i = 5, j = 9):

P59 = 0 + 5 + 4 + 0 = 9

N59 = 0 + 0 + 0 + 0 = 0

D59 = P59/N59 = 9/0 - Деление на ноль, отбрасываем

P95 = 0 + 0 + 0 + 0 = 0

N95 = 0 + 5 + 4 + 0 = 9

D95 = N95/P95 = 0/9 - inf - принимаем

Рассмотрим альтернативы 6 и 7 (i = 6, j = 7):

P67 = 0 + 5 + 0 + 3 = 8

N67 = 5 + 0 + 0 + 0 = 5

D67 = P67/N67 = 8/5 = 1.6 > 1 - принимаем

P76 = 5 + 0 + 0 + 0 = 5

N76 = 0 + 5 + 0 + 3 = 8

D76 = N76/P76 = 5/8 = 0.6 < 1 - отбрасываем

Рассмотрим альтернативы 6 и 8 (i = 6, j = 8):

P68 = 0 + 0 + 4 + 3 = 7

N68 = 5 + 0 + 0 + 0 = 5

D68 = P68/N68 = 7/5 = 1.4 > 1 - принимаем

P86 = 5 + 0 + 0 + 0 = 5

N86 = 0 + 0 + 4 + 3 = 7

D86 = N86/P86 = 5/7 = 0.7 < 1 - отбрасываем

Рассмотрим альтернативы 6 и 9 (i = 6, j = 9):

P69 = 0 + 5 + 4 + 0 = 9

N69 = 5 + 0 + 0 + 0 = 5

D69 = P69/N69 = 9/5 = 1.8 > 1 - принимаем

P96 = 5 + 0 + 0 + 0 = 5

N96 = 0 + 5 + 4 + 0 = 9

D96 = N96/P96 = 5/9 = 0.6 < 1 - отбрасываем

Рассмотрим альтернативы 7 и 8 (i = 7, j = 8):

P78 = 5 + 0 + 4 + 0 = 9

N78 = 0 + 5 + 0 + 0 = 5

D78 = P78/N78 = 9/5 = 1.8 > 1 - принимаем

P87 = 0 + 5 + 0 + 0 = 5

N87 = 5 + 0 + 4 + 0 = 9

D87 = N87/P87 = 5/9 = 0.6 < 1 - отбрасываем

Рассмотрим альтернативы 7 и 9 (i = 7, j = 9):

P79 = 5 + 0 + 4 + 0 = 9

N79 = 0 + 0 + 0 + 3 = 3

D79 = P79/N79 = 9/3 = 3.0 > 1 - принимаем

P97 = 0 + 0 + 0 + 3 = 3

N97 = 5 + 0 + 4 + 0 = 9

D97 = N97/P97 = 3/9 = 0.3 < 1 - отбрасываем

Рассмотрим альтернативы 8 и 9 (i = 8, j = 9):

P89 = 0 + 5 + 4 + 0 = 9

N89 = 0 + 0 + 0 + 3 = 3

D89 = P89/N89 = 9/3 = 3.0 > 1 - принимаем

P98 = 0 + 0 + 0 + 3 = 3

N98 = 0 + 5 + 4 + 0 = 9

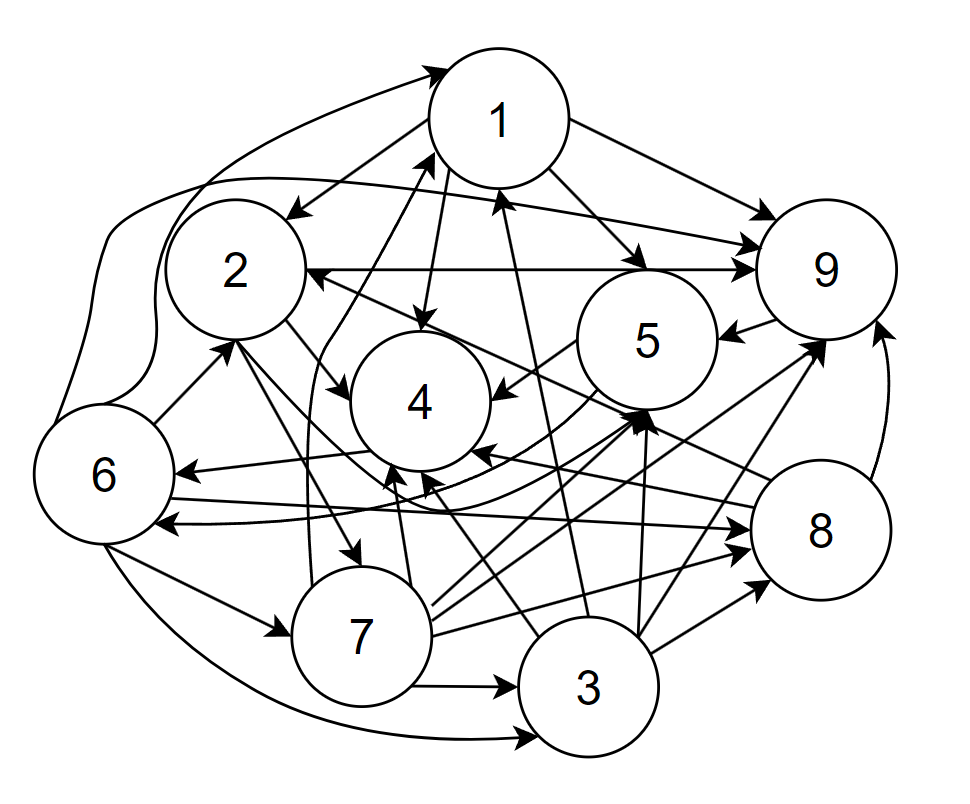
D98 = N98/P98 = 3/9 = 0.3 < 1 – отбрасываем

Составлена матрица предпочтений с внесенными и принятыми значениями D.

*Таблица 2.2.1 – Полная матрица предпочтений альтернатив.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | x | Inf |  | 1.1 | Inf |  |  |  | 3.0 |
| 2 |  | x |  | 1.1 | 1.7 |  | 1.2 |  | 4.7 |
| 3 | ­­­­­1.7 | Inf | x | 1.1 | 1.7 |  |  | 1.7 | 4.7 |
| 4 |  |  |  | x |  | 1.2 |  |  |  |
| 5 |  |  |  | 1.8 | x | 1.2 |  |  |  |
| 6 | 1.4 | 1.4 | 1.4 |  |  | x | 1.6 | 1.4 | 1.8 |
| 7 | 1.8 |  | 1.4 | 1.1 | 1.1 |  | x | 1.8 | 3.0 |
| 8 |  | inf |  | 1.1 | inf |  |  | x | 3.0 |
| 9 |  |  |  |  | inf |  |  |  | x |

По матрице построен граф предпочтений (Рисунок 2.2.1).



**Рисунок 2.2.1 – Вид графа предпочтений**

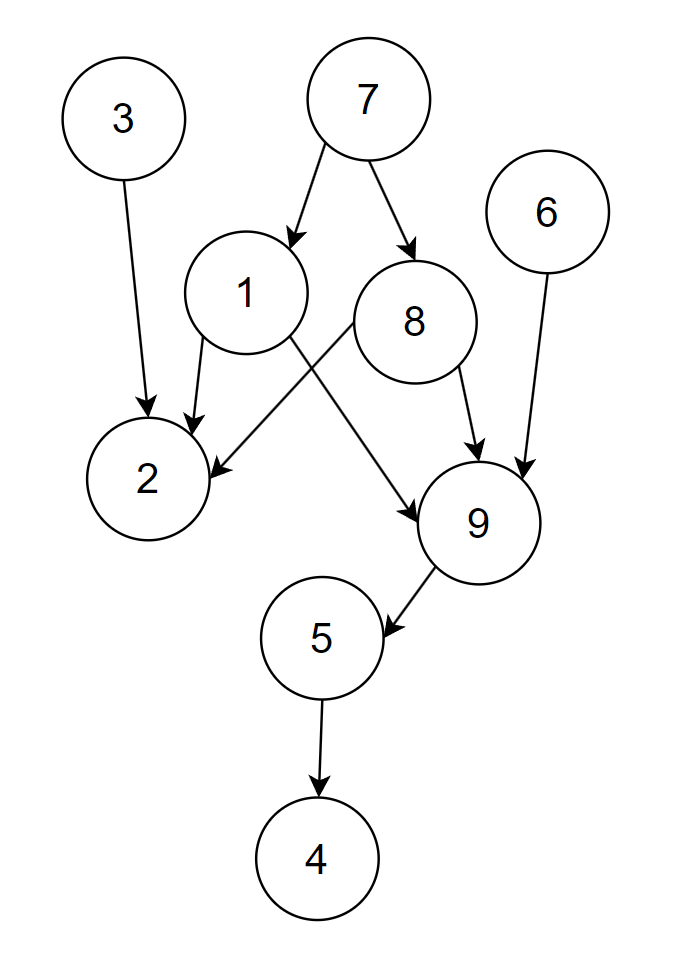
Назначен порог отбора предпочтений C = 1.8 (это соответствует тому, что учитываются только более сильные связи в графе).

Таким образом, матрица разрежается. В ней остаются только самые сильные связи (Таблица 2.2.2).

*Таблица 2.2.2* **–** *Матрица предпочтений проектов, при пороге С=1.8*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | x | Inf |  |  | Inf |  |  |  | 3.0 |
| 2 |  | x |  |  |  |  |  |  | 4.7 |
| 3 | ­­­­­ | Inf | x |  |  |  |  |  | 4.7 |
| 4 |  |  |  | x |  |  |  |  |  |
| 5 |  |  |  | 1.8 | x |  |  |  |  |
| 6 |  |  |  |  |  | x |  |  | 1.8 |
| 7 | 1.8 |  |  |  |  |  | x | 1.8 | 3.0 |
| 8 |  | inf |  |  | inf |  |  | x | 3.0 |
| 9 |  |  |  |  | inf |  |  |  | x |

По этой матрице построен граф предпочтений (Рисунок 2.2.2).

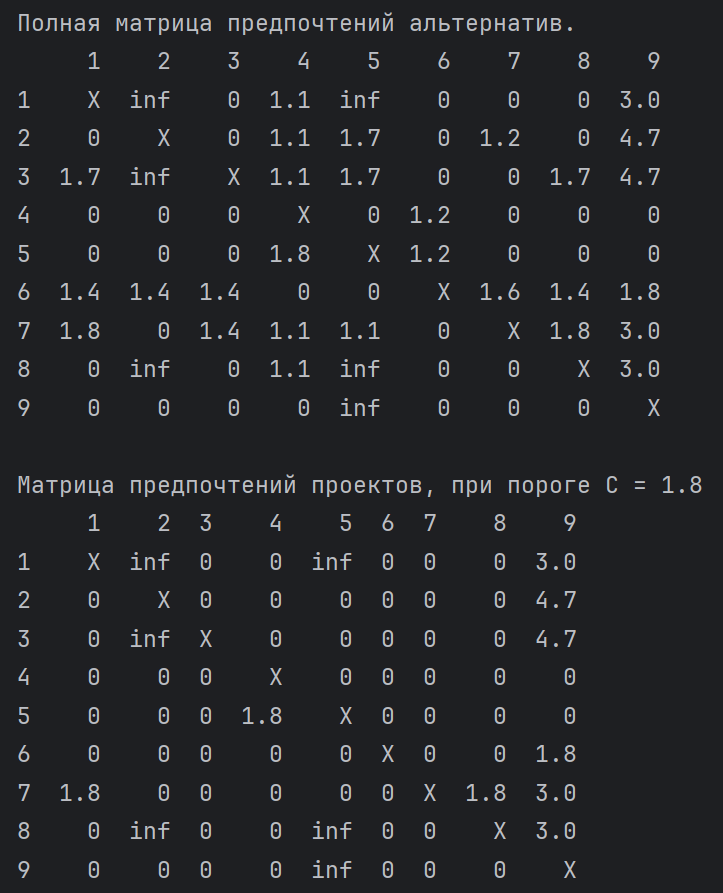


**Рисунок 2.2.2 – Вид графа предпочтений для случая порога принятия решений C = 2.4**

Петель в графе нет, при этом граф остался целостным.

Решение говорит нам о том, что лучшие проекты – 8, 7, 1. Потом 3 за ним 9, 6, 5 и 2, а самые худший – это 4

## **2.3 Результат работы программы**



**Рисунок 2.3.1 – Результат работы программы. Вывод матрицы предпочтений.**

## **2.4 Вывод по методу Электра II**

Метод Электра II успешно применен для анализа и ранжирования альтернатив на основе заданных критериев, что позволило учесть компромиссы между ними. Его преимущества включают гибкость и прозрачность, однако метод требует значительных вычислений и чувствителен к выбору весов критериев.

3 МЕТОД АНАЛИЗА ИЕРАРХИЙ

Метод анализа иерархий (МАИ) является мощным инструментом, применяемым для решения задач, связанных с многокритериальной оценкой и принятием решений. Он используется для упорядочивания множества альтернатив по определённым критериям, что позволяет принять обоснованное решение в условиях неопределённости и сложности. МАИ основывается на сравнении элементов пары, что помогает учитывать все возможные предпочтения и важность факторов для каждого варианта.

Этот метод широко используется в различных областях, таких как управление проектами, бизнес-анализ, выбор оптимальных стратегий, а также в разработке систем поддержки принятия решений. Например, в задачах выбора наилучшей модели автомобиля, оптимизации производственных процессов, распределении ресурсов или планировании.

Применение МАИ позволяет структурировать и формализовать процесс принятия решений, минимизируя субъективность и учитывая различные аспекты, что делает его полезным инструментом в задачах с множеством критериев, таких как выбор поставщика, оценка финансовых показателей или даже в личных предпочтениях.

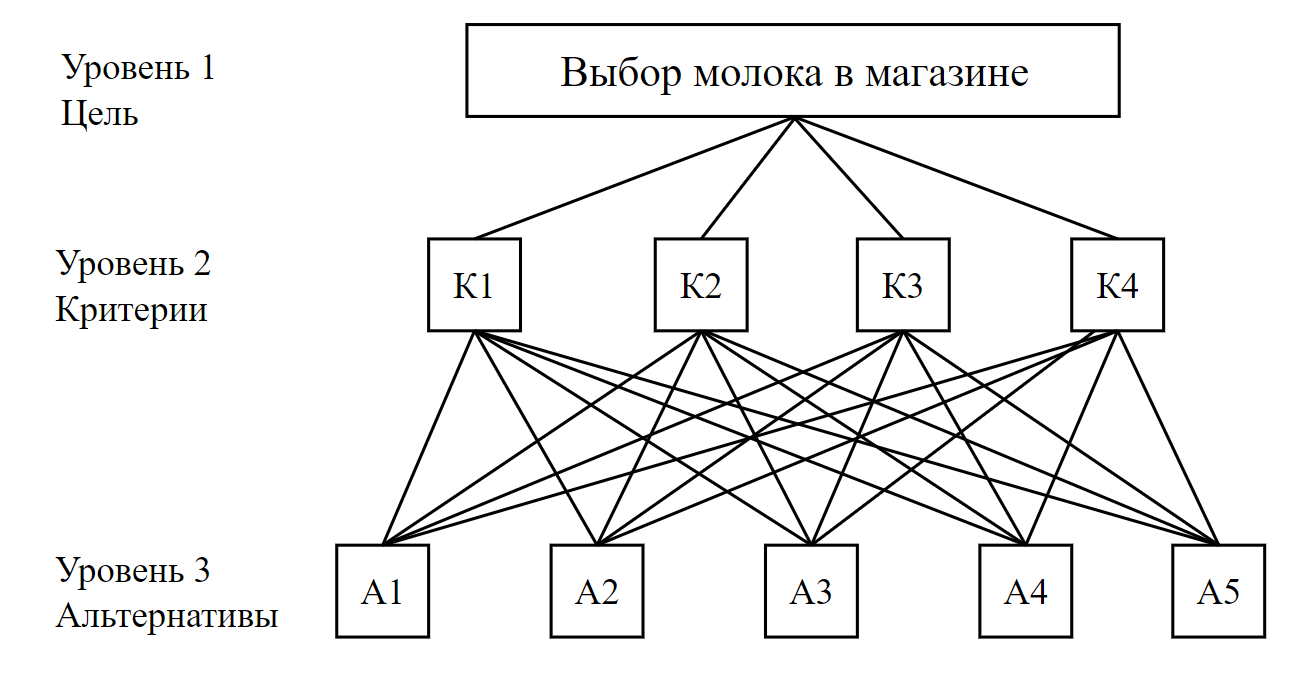
## **3.1 Постановка задачи**

Задача практической работы: выбор молока в магазине.

## **3.2 Представление проблемы в виде иерархии**

Первый этап – представление проблемы в виде иерархии или сети. В простейшем случае, иерархия строится, начиная с цели, которая помещается в вершину иерархии. Через промежуточные уровни, на которых располагаются критерии и от которых зависят последующие уровни, к самому низкому уровню, который содержит перечень альтернатив.

Иерархия считается полной, если каждый элемент заданного уровня является критерием для всех элементов нижнего уровня



**Рисунок 3.2.1 – Полная доминантная иерархия.**

Критерии:

К 1 – Цена;

К 2 – Вес;

К 3 – Калорийность;

К 4 – Срок хранения;

Альтернативы:

А 1 - Простоквашино;

А 2 - Просто;

А 3 - Сарафаново;

А 4 – Зелёная линия;

А 5 – Искренне ваш.

## **3.3 Установка приоритетов критериев**

После иерархического представления задачи установлены приоритеты критериев и оценена каждая из альтернатив по критериям, определена наиболее важная их них. В методе анализа иерархий элементы сравниваются попарно по отношению к их влиянию на общую для них характеристику. Парные сравнения приводят к записи характеристик сравнений в виде квадратной таблицы чисел, которая называется матрицей. Для облегчения работы введена шкала относительной важности (Таблица 1).

*Таблица.3.3.1 – Шкала относительной важности.*

|  |  |  |
| --- | --- | --- |
| Интенсивность  относительной  важности | Определение | Объяснение |
| 1 | Равная важность | Равный вклад двух критериев в цель. |
| 3 | Слабое превосходство | Дают легкое превосходство одной альтернативы над другой |
| 5 | Умеренное превосходство | Опыт и суждения дают умеренное превосходство |
| 7 | Сильное превосходство | Одному из критериев дается настолько сильное предпочтение. |
| 9 | Абсолютное превосходство | Очевидность превосходства одного критерия над другим |
| 2,4,6,8 | Промежуточные решения между двумя соседними суждениями | Применяется в компромиссных случаях |

Шкала содержит соответствующие обратные значения.

## **3.4 Синтез приоритетов**

После построения иерархии и определения величин парных субъективных суждений следует этап, на котором иерархическая декомпозиция и относительные суждения объединяются для получения осмысленного решения многокритериальной задачи принятия решений. Из групп парных сравнений формируется набор локальных критериев, которые выражают относительное влияние элементов на элемент, расположенный на уровне выше. Составлена обратно симметричная матрица для парного сравнения критериев (Таблица 2).

*Таблица 3.4.1 – Матрица парного сравнения критериев.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Цель | К 1 | К 2 | К 3 | К 4 | Vi | W2i |
| К 1 | 1 | 1 | 3 | 5 | 1.968 | 0.391 |
| К 2 | 1 | 1 | 3 | 5 | 1.968 | 0.391 |
| К 3 | 1/3 | 1/3 | 1 | 3 | 0.760 | 0.151 |
| К 4 | 1/5 | 1/5 | 1/3 | 1 | 0.340 | 0.067 |
| ∑Vi | | | | | 5.036 |

Для определения относительной ценности каждого элемента необходимо найти геометрическое среднее и с этой целью перемножить n элементов каждой строки и из полученного результата извлечь корни n-й степени (размерность матрицы n=5).

Строка № 1

V1= (1х1х3х5)1/4 = 1.968;

Строка № 2

V2= (1х1х3х5)1/4 = 1.968;

Строка № 3

V3=(1/3x1/3x1x3)1/4 = 0.760;

Строка № 4

V4=(1/5x1/5x1/3x1)1/4 = 0.340;

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑Vi.

∑Vi = 1.968 + 1.968 + 0.760 + 0.340 = 5.036

Найдена важность приоритетов W2i, для этого каждое из чисел Vi разделено на ∑Vi.

***Строка № 1***

W21= 1.968 / 5.036 = 0.391;

Строка № 2

W22= 1.968 / 5.036 = 0.391;

***Строка № 3***

W23= 0.760 / 5.036 = 0.151;

***Строка № 4***

W24= 0.340 / 5.036 = 0.067;

В результате получен вектор приоритетов:

W2i = (0.391; 0.391; 0.151; 0.067), где индекс 2 означает, что вектор приоритетов относится ко второму уровню иерархии.

К 1 – Цена (Таблица 3);

*Таблица 3.4.2 – Матрица сравнения по критерию 1.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| К1 | А1 | А2 | А3 | А4 | А5 | VК1Y | W3К1Y |
| А1 | 1 | 1/5 | 1/7 | 1/7 | 1/9 | 0.214 | 0.028 |
| А2 | 5 | 1 | 1/5 | 1/5 | 1/7 | 0.491 | 0.065 |
| А3 | 7 | 5 | 1 | 1 | 1/3 | 1.635 | 0.217 |
| А4 | 7 | 5 | 1 | 1 | 1/3 | 1.635 | 0.217 |
| А5 | 9 | 7 | 3 | 3 | 1 | 3.554 | 0.472 |
| ∑VК1Y | | | | | | 7.529 |

Определена относительная ценность каждого элемента.

Строка № 1

VК11=(1x1/5x1/7x1/7x1/9)1/5= 0.214;

Строка № 2

VК12=(5x1x1/5x1/5x1/5x1/7)1/5= 0.491;

Строка № 3

VК13=(7x5x1x1x1/3)1/5= 1.635;

Строка № 4

VК14=(7x5x1x1x1/3)1/5= 1.635;

Строка № 5

VК15=(9x7x3x3x1)1/5= 3.554.

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑VK1Y.

∑VК1Y = 0.214 + 0.491 + 1.635 + 1.635 + 3.554 = 7.529.

Найдена важность приоритетов W3К1Y, для этого каждое из чисел VK1Y разделено на ∑VK1Y.

Строка № 1

W3К11= 0.214 / 7.529 = 0.028;

Строка № 2

W3К12= 0.491 / 7.529 = 0.065;

Строка № 3

W3К13= 1.635 / 7.529 = 0.217;

Строка № 4

W3К14= 1.635 / 7.529 = 0.217;

Строка № 5

W3К15= 3.554 / 7.529 = 0.472;

В результате получаем вектор приоритетов:

W3К1Y = (0.028; 0.065; 0.217; 0.217; 0.472)

где индекс 3 означает, что вектор приоритетов относится к третьему уровню иерархии критерия К1.

К 2 – Вес (Таблица 4);

*Таблица 3.4.3. – Матрица сравнения по критерию 2.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| К2 | А1 | А2 | А3 | А4 | А5 | VК2Y | W3К2Y |
| А1 | 1 | 5 | 1 | 5 | 7 | 2.809 | 0.389 |
| А2 | 1/5 | 1 | 1/5 | 1 | 3 | 0.654 | 0.091 |
| А3 | 1 | 5 | 1 | 5 | 7 | 2.809 | 0.389 |
| А4 | 1/5 | 1 | 1/5 | 1 | 3 | 0.654 | 0.091 |
| А5 | 1/7 | 1/3 | 1/7 | 1/3 | 1 | 0.296 | 0.041 |
| ∑VК2Y | | | | | | 7.223 |  |

Определена относительная ценность каждого элемента.

Строка № 1

VК21=(1x5x1x5x7)1/5= 2.809;

Строка № 2

VК22=(1/5x1x1/5x1x3)1/5=0.654;

Строка № 3

VК23=(1x5x1x5x7)1/5= 2.809;

Строка № 4

VК24=(1/5x1x1/5x1x3)1/5= 0.654;

Строка № 5

VК25=(1/7x1/3x1/7x1/3x1)1/5= 0.296.

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑VK2Y.

∑VК2Y = 2.809 + 0.654 + 2.809 + 0.654 + 0.296 = 7.223.

Найдена важность приоритетов W3К2Y, для этого каждое из чисел VK2Y разделено на ∑VK2Y.

Строка № 1

W3К21= 2.809 / 7.223 = 0.389;

Строка № 2

W3К22= 0.654 / 7.223 = 0.091;

Строка № 3

W3К23= 2.809 / 7.223 = 0.389;

Строка № 4

W3К24= 0.654 / 7.223 = 0.091;

Строка № 5

W3К25= 0.296 / 7.223 = 0.041.

В результате получаем вектор приоритетов:

W3К2Y = (0.389; 0.091; 0.389; 0.091; 0.041),

где индекс 3 означает, что вектор приоритетов относится к третьему уровню иерархии критерия К2.

К 3 – Калорийность (Таблица 5);

*Таблица 3.4.4 – Матрица сравнения по критерию 5.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| К3 | А1 | А2 | А3 | А4 | А5 | VК3Y | W3К3Y |
| А1 | 1 | 5 | 5 | 5 | 1/3 | 2.108 | 0.278 |
| А2 | 1/5 | 1 | 1 | 1 | 1/7 | 0.491 | 0.065 |
| А3 | 1/5 | 1 | 1 | 1 | 1/7 | 0.491 | 0.065 |
| А4 | 1/5 | 1 | 1 | 1 | 1/7 | 0.491 | 0.065 |
| А5 | 3 | 7 | 7 | 7 | 1 | 4.004 | 0.528 |
| VК35 | | | | | | 7.586 |

Определена относительная ценность каждого элемента.

Строка № 1

VК31= (1x5x5x5x1/3)1/5= 2.108;

Строка № 2

VК32= (1/5x1x1x1x1/7)1/5= 0.491;

Строка № 3

VК33= (1/5x1x1x1x1/7)1/5= 0.491;

Строка № 4

VК34= (1/5x1x1x1x1/7)1/5= 0.491;

Строка № 5

VК35= (3x7x7x7x1)1/5= 4.004.

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑VK3Y.

∑VК3Y = 2.108 + 0.491 + 0.491 + 0.491 + 4.004 = 7.586.

Найдена важность приоритетов W3К2Y, для этого каждое из чисел VK2Y разделено на ∑VK2Y.

Строка № 1

W3К31= 2.108 / 7.586 = 0.278;

Строка № 2

W3К32= 0.491 / 7.586 = 0.065;

Строка № 3

W3К33= 0.491 / 7.586 = 0.065;

Строка № 4

W3К34= 0.491 / 7.586 = 0.065;

Строка № 5

W3К35= 4.004 / 7.586 = 0.528;

В результате получаем вектор приоритетов:

W3К3Y = (0.278; 0.065; 0.065; 0.065; 0.528),

где индекс 3 означает, что вектор приоритетов относится к третьему уровню иерархии критерия К3.

К 4 – срок хранения (Таблица 6);

*Таблица 3.4.5 – Матрица сравнения по критерию 4.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| К4 | А1 | А2 | А3 | А4 | А5 | VК4Y | W3К4Y |
| А1 | 1 | 9 | 1 | 9 | 9 | 3.737 | 0.423 |
| А2 | 1/9 | 1 | 1/9 | 1 | 5 | 0.573 | 0.065 |
| А3 | 1 | 9 | 1 | 9 | 9 | 3.737 | 0.423 |
| А4 | 1/9 | 1 | 1/9 | 1 | 5 | 0.573 | 0.065 |
| А5 | 1/9 | 1/5 | 1/9 | 1/5 | 1 | 0.218 | 0.025 |
| ∑VК4Y | | | | | | 8.838 |

Определена относительная ценность каждого элемента.

Строка № 1

VК41=(1x9x1x9x9)1/5 = 3.737;

Строка № 2

VК42=(1/9x1x1/9x1x5)1/5 = 0.573;

Строка № 3

VК43=(x9x1x9x9)1/5 = 3.737;

Строка № 4

VК44=(1/9x1x1/9x1x5)1/5 = 0.573;

Строка № 5

VК45=(1/9x1/5x1/9x1/5x1)1/5 = 0.218.

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑VK4Y.

∑VК4Y = 3.737 + 0.573 + 3.737 + 0.573 + 0.218 = 8.838.

Найдена важность приоритетов W3К4Y, для этого каждое из чисел VK4Y разделено на ∑VK4Y.

Строка № 1

W3К41= 3.737 / 8.838 = 0.423;

Строка № 2

W3К42= 0.573 / 8.838 = 0.065;

Строка № 3

W3К43= 3.737 / 8.838 = 0.423;

Строка № 4

W3К44= 0.573 / 8.838 = 0.065;

Строка № 5

W3К45= 0.218 / 8.838 = 0.025.

В результате получаем вектор приоритетов:

W3К4Y = (0.423; 0.065; 0.423; 0.065; 0.025),

где индекс 3 означает, что вектор приоритетов относится к третьему уровню иерархии критерия К4.

## **3.5 Согласованность локальных приоритетов**

Любая матрица суждений в общем случае не согласована, так как суждения отражают субъективные мнения ЛПР, а сравнение элементов, которые имеют количественные эквиваленты, может быть несогласованным из-за присутствия погрешности при проведении измерений. Совершенной согласованности парных сравнений даже в идеальном случае на практике достичь трудно. Нужен способ оценки степени согласованности при решении конкретной задачи.

Метод анализа иерархий дает возможность провести такую оценку.

Вместе с матрицей парных сравнений есть мера оценки степени отклонения от согласованности. Когда такие отклонения превышают установленные пределы тем, кто проводит решение задачи, необходимо их пересмотреть.

В таблице приведены средние значения индекса случайной согласованности (СИ) для случайных матриц суждений разного порядка.

В нашей задаче размерность матрицы n=5, тогда среднее значение индекса случайной согласованности СИ = 1,12.

Определены индекс согласованности и отношение согласованности для матрицы «Выбор лучшего отеля» (Таблица 8).

*Таблица 3.5.1 – Матрица «Выбор молока в магазине».*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Цель | К 1 | К 2 | К 3 | К 4 | W2i |
| К 1 | 1 | 1 | 3 | 5 | 0.391 |
| К 2 | 1 | 1 | 3 | 5 | 0.391 |
| К 3 | 1/3 | 1/3 | 1 | 3 | 0.151 |
| К 4 | 1/5 | 1/5 | 1/3 | 1 | 0.067 |

Определена сумма каждого столбца матрицы суждений.

S1 = 1 + 1 + 0.333 + 0.2 = 2.533;

S2 = 1 + 1 + 0.333 + 0.2 = 2.533;

S3 = 3 + 3 + 1 + 0.333 = 7.333;

S4 = 5 + 5 + 3 + 1 = 14;

Полученный результат умножен на компоненту нормализованного вектора приоритетов, т.е. сумму суждений первого столбца на первую компоненту, сумму суждений второго столбца - на вторую и т.д.

Р1 = S1 х W21 = 2.533 × 0.391 = 0.99;

Р2 = S2 х W22 = 2.533 × 0.391 = 0.99;

Р3 = S3 х W23 = 7.333 × 0.151 = 1.11;

Р4 = S4 х W24 = 14 × 0.067 = 0.94;

Сумма чисел Рj отражает пропорциональность предпочтений, чем ближе эта величина к n (числу объектов и видов действия в матрице парных сравнений), тем более согласованны суждения.

λmax = Р1 + Р2 + Р3 + Р4 = 4.03.

Отклонение от согласованности выражается индексом согласованности.

ИС = (λmax - n)/(n - 1) = (4.03 - 4)/(4 - 1) = 0.0105.

Отношение индекса согласованности ИС к среднему значению случайного индекса согласованности СИ называется отношением согласованности ОС.

ОС = ИС/СИ = 0.0105/0.9 = 0.012.

Значение ОС меньше или равное 0.10 считается приемлемым, значит матрица «Выбор лучшего спортсмена» согласована.

Определнены индекс согласованности и отношение согласованности для матрицы К 1 – цена (Таблица 9).

*Таблица 3.5.2 – Матрица сравнения по критерию 1.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| К1 | А1 | А2 | А3 | А4 | А5 | W3К1Y |
| А1 | 1 | 1/5 | 1/7 | 1/7 | 1/9 | 0.028 |
| А2 | 5 | 1 | 1/5 | 1/5 | 1/7 | 0.065 |
| А3 | 7 | 5 | 1 | 1 | 1/3 | 0.217 |
| А4 | 7 | 5 | 1 | 1 | 1/3 | 0.217 |
| А5 | 9 | 7 | 3 | 3 | 1 | 0.472 |

Определяется сумма каждого столбца матрицы суждений.

S1К1 = 1 + 5 + 7 + 7 + 9 = 29;

S2 К1 = 0.2 + 1 + 5 + 5 + 7 = 18.2;

S3 К1 = 0.143 + 0.2 + 1 + 1 + 3 = 5.343;

S4 К1 = 0.143 + 0.2 + 1 + 1 + 3 = 5.343;

S5 К1 =0.111 + 0.143 + 0.333 + 0.333 + 1 = 1.921.

Затем полученный результат умножен на компоненту нормализованного вектора приоритетов.

Р1 К1 = S1 х W3К11 = 29.000 × 0.028 = 0.83;

Р2 К1 = S2 х W3К12 = 18.200 × 0.065 = 1.19;

Р3 К1 = S3 х W3К13 = 5.343 × 0.217 = 1.16;

Р4 К1 = S1 х W3К14 = 5.343 × 0.217 = 1.16;

Р5 К1 = S1 х W3К15 = 1.921 × 0.472 = 0.91.

Найдена пропорциональность предпочтений.

λmax К1 = Р1К1 + Р2К1 + Р3К1 + Р4К1 + Р5К1 = 5.24.

Отклонение от согласованности выражается индексом согласованности.

ИС К1 = (λmax К1 - n)/(n - 1) = (5.24 - 5)/(5 - 1) = 0.0600.

Найдено отношение согласованности ОС.

ОС К1 = ИС/СИ = 0.0600/1.12 = 0.054.

Значение ОС меньше или равное 0.10 считается приемлемым, значит матрица К 1 (цена дня проживания) согласована.

Определены индекс согласованности и отношение согласованности для матрицы К 2 – вес (Таблица 10).

*Таблица 3.5.3 – Матрица сравнения по критерию 2.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| К2 | А1 | А2 | А3 | А4 | А5 | W3К2Y |
| А1 | 1 | 5 | 1 | 5 | 7 | 0.389 |
| А2 | 1/5 | 1 | 1/5 | 1 | 3 | 0.091 |
| А3 | 1 | 5 | 1 | 5 | 7 | 0.389 |
| А4 | 1/5 | 1 | 1/5 | 1 | 3 | 0.091 |
| А5 | 1/7 | 1/3 | 1/7 | 1/3 | 1 | 0.041 |

Определена сумма каждого столбца матрицы суждений.

S1К2 = 1 + 0.2 + 1 + 0.2 + 0.143 = 2.543;

S2 К2 = 5 + 1 + 5 + 1 + 0.333 = 12.333;

S3 К2 = 1 + 0.2 + 1 + 0.2 + 0.143 = 2.543;

S4 К2 = 5 + 1 + 5 + 1 + 0.333 = 12.333;

S5 К2 = 7 + 3 + 7 + 3 + 1 = 21.

Затем полученный результат умножен на компоненту нормализованного вектора приоритетов.

Р1 К2 = S1 х W3 К21 = 2.543 × 0.389 = 0.99;

Р2 К2 = S2 х W3 К22 = 12.333 × 0.091 = 1.12;

Р3 К2 = S3 х W3 К23 = 2.543 × 0.389 = 0.99;

Р4 К2 = S4 х W3 К24 = 12.333 × 0.091 = 1.12;

Р5 К2 = S5 х W3 К25 = 21.000 × 0.041 = 0.86.

Найдена пропорциональность предпочтений.

λmax К2 = Р1К2 + Р2К2 + Р3К2 + Р4К2 + Р5К2 = 5.07.

Отклонение от согласованности выражается индексом согласованности.

ИС К2 = (λmax К2 - n)/(n - 1) = (5.07 - 5)/(5 - 1) = 0.0182.

Найдено отношение согласованности ОС.

ОС К2 = ИС/СИ = 0.0182/1.12 = 0.016.

Значение ОС меньше или равное 0.10 считается приемлемым, значит матрица К 2 (количество звезд) согласована.

Определены индекс согласованности и отношение согласованности для матрицы К 3 – калорийность (Таблица 11).

*Таблица 3.5.4 – Матрица сравнения по критерию 3.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| К3 | А1 | А2 | А3 | А4 | А5 | W3К3Y |
| А1 | 1 | 5 | 5 | 5 | 1/3 | 0.278 |
| А2 | 1/5 | 1 | 1 | 1 | 1/7 | 0.065 |
| А3 | 1/5 | 1 | 1 | 1 | 1/7 | 0.065 |
| А4 | 1/5 | 1 | 1 | 1 | 1/7 | 0.065 |
| А5 | 3 | 7 | 7 | 7 | 1 | 0.528 |

Определена сумма каждого столбца матрицы суждений.

S1К3 = 1 + 0.2 + 0.2 + 0.2 + 3 = 4.600;

S2 К3 = 5 + 1 + 1 + 1 + 7 = 15;

S3 К3 = 5 + 1 + 1 + 1 + 7 = 15;

S4 К3 = 5 + 1 + 1 + 1 + 7 = 15;

S5 К3 = 0.333 + 0.143 + 0.143 + 0.143 + 1 = 1.762.

Затем полученный результат умножен на компоненту нормализованного вектора приоритетов.

Р1 К3 = S1 х W3 К31 = 4.6 × 0.278 = 1.28;

Р2 К3 = S2 х W3 К32 = 15 × 0.065 = 0.97;

Р3 К3 = S3 х W3 К33 = 15 × 0.065 = 0.97;

Р4 К3 = S4 х W3 К34 = 15 × 0.065 = 0.97;

Р5 К3 = S5 х W3 К35 = 1.762 × 0.528 = 0.93.

Найдем пропорциональность предпочтений.

λmax К3 = Р1К3 + Р2К3 + Р3К3 + Р4К3 + Р5К3 = 5.12.

Отклонение от согласованности выражается индексом согласованности.

ИС К3 = (λmax К3 - n)/(n - 1) = (5.12 - 5)/(5 - 1) = 0.0305.

Найдено отношение согласованности ОС.

ОС К3 = ИС/СИ = 0.0305/1.12 = 0.027.

Значение ОС меньше или равное 0.10 считается приемлемым, значит матрица К 3 (рейтинг по отзывам пользователей) согласована.

Определены индекс согласованности и отношение согласованности для матрицы К 4 – срок хранения (Таблица 12).

*Таблица 3.5.5 – Матрица сравнения по критерию 4.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| К4 | А1 | А2 | А3 | А4 | А5 | W3К4Y |
| А1 | 1 | 9 | 1 | 9 | 9 | 0.423 |
| А2 | 1/9 | 1 | 1/9 | 1 | 5 | 0.065 |
| А3 | 1 | 9 | 1 | 9 | 9 | 0.423 |
| А4 | 1/9 | 1 | 1/9 | 1 | 5 | 0.065 |
| А5 | 1/9 | 1/5 | 1/9 | 1/5 | 1 | 0.025 |

Определена сумма каждого столбца матрицы суждений.

S1К4 = 1 + 0.111 + 1 + 0.111 + 0.111 = 2.333;

S2К4 = 9 + 1 + 9 + 1 + 0.2 = 20.2;

S3К4 = 1 + 0.111 + 1 + 0.111 + 0.111 = 2.333;

S4К4 = 9 + 1 + 9 + 1 + 0.2 = 20.2;

S5К4 = 9 + 5 + 9 + 5 + 1 = 29.

Затем полученный результат умножен на компоненту нормализованного вектора приоритетов.

Р1К4 = S1 х W3 К41 = 2.333 × 0.423 = 0.99;

Р2К4 = S2 х W3 К42 = 20.200 × 0.065 = 1.31;

Р3К4 = S3 х W3 К43 = 2.333 × 0.423 = 0.99;

Р4К4 = S4 х W3 К44 = 20.200 × 0.065 = 1.31;

Р5К4 = S5 х W3 К45 = 29.000 × 0.025 = 0.72.

Найдена пропорциональность предпочтений.

λmax К4 = Р1К4 + Р2К4 + Р3К4 + Р4К4 + Р5К4 = 5.31.

Отклонение от согласованности выражается индексом согласованности.

ИС К4 = (λmax К4 - n)/(n - 1) = (5.31 - 5)/(5 - 1) = 0.0769.

Найдено отношение согласованности ОС.

ОС К4 = ИС/СИ = 0.0769/1.12 = 0.069.

Значение ОС меньше или равное 0,10 считается приемлемым, значит матрица К 4 (удаленность от ближайшей станции метро) согласована.

## **3.6 Синтез альтернатив**

Векторы приоритетов и отношения согласованности определяются для всех матриц суждений, начиная со второго уровня.

Для определения приоритетов альтернатив локальные приоритеты умножены на приоритет соответствующего критерия на высшем уровне и найдены суммы по каждому элементу в соответствии с критериями, на которые воздействует этот элемент.

W2i = (0.391; 0.391; 0.151; 0.067);

W3К1Y = (0.028; 0.065; 0.217; 0.217; 0.472);

W3К2Y = (0.389; 0.091; 0.389; 0.091; 0.041);

W3К3Y = (0.278; 0.065; 0.065; 0.065; 0.528);

W3К4Y (0.423; 0.065; 0.423; 0.065; 0.025);

Приоритеты альтернатив получены следующим образом:

W1 = W21 х W3К11 + W22 х W3К21 + W23 х W3К31 + W24 х W3К41 = 0.391 x 0.028

+ 0.391 x 0.389 +0.151 x 0.278 + 0.067 x 0.423 = 0.233.

W2 = W21 х W3К12 + W22 х W3К22 + W23 х W3К32 + W24 х W3К42 = 0.391 × 0.065 + 0.391 × 0.091 + 0.151 × 0.065 + 0.067 × 0.065 = 0.075.

W3 = W21 х W3К13 + W22 х W3К23 + W23 х W3К33 + W24 х W3К43 = 0.391 × 0.217 + 0.391 × 0.389 + 0.151 × 0.065 + 0.067 × 0.423 = 0.275

W4 = W21 х W3К14 + W22 х W3К24 + W23 х W3К34 + W24 х W3К44 = 0.391 × 0.217 + 0.391 × 0.091 + 0.151 × 0.065 + 0.067 × 0.065 = 0.135.

W5 = W21 х W3К15 + W22 х W3К25 + W23 х W3К35 + W24 х W3К45 = 0.391 × 0.472 + 0.391 × 0.041 + 0.151 × 0.528 + 0.067 × 0.025 = 0.282.

Таким образом, приоритеты альтернатив равны:

альтернатива А1 (Сарафаново) - W1 приоритет равен = 0.233;

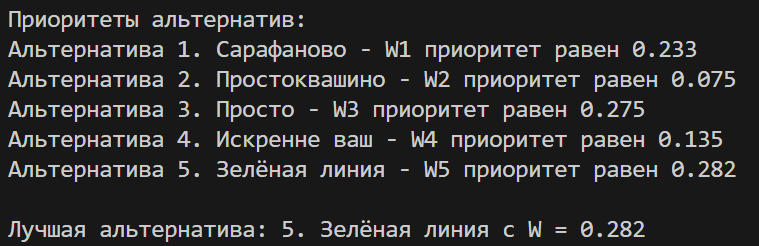
альтернатива А2 (Простоквашино)- W2 приоритет равен = 0.075;

альтернатива А3 (Просто) - W3 приоритет равен = 0.275;

альтернатива А4 (Искренне ваш) – W4 приоритет равен = 0.135;

альтернатива А5 (Зелёная линия) - W5 приоритет равен = 0.282.

## **3.7 Результаты работы программы**



**Рисунок 3.7.1 – Вывод программы**

## **Вывод по методу МАИ**

В ходе работы был использован метод анализа иерархий (МАИ) для решения задачи многокритериальной оценки и принятия решений. Метод был применён для сравнения альтернатив по различным критериям, что позволило выявить наиболее предпочтительные варианты. В результате применения МАИ были получены приоритеты альтернатив, что обеспечило объективное и структурированное решение.

Преимущества метода МАИ включают его способность систематизировать сложные задачи с множеством критериев, чётко формализовать процесс принятия решений и снизить влияние субъективных факторов. МАИ позволяет учитывать разные взгляды и предпочтения при принятии решения, что делает его универсальным инструментом для различных областей: от бизнеса до научных исследований.

Однако метод имеет и недостатки. Он может быть чувствителен к ошибкам в оценке сравнений, особенно когда различия между альтернативами незначительны. Кроме того, для корректной работы требуется точность и внимательность при составлении матрицы суждений, что может быть сложным для неопытных пользователей. Также важно учитывать, что при большом количестве критериев и альтернатив МАИ может стать вычислительно дорогим и трудоемким.

**4 ГРАФИЧЕСКИЙ МЕТОД**

Графический метод в теории принятия решений является одним из наиболее наглядных и простых инструментов для анализа и выбора оптимального решения в условиях ограниченного числа переменных. Этот метод широко применяется в задачах линейного программирования, экономического анализа, управленческих и инженерных задачах, где требуется найти максимум или минимум целевой функции при заданных ограничениях.

Основное преимущество графического метода заключается в его визуальной интерпретации: решение отображается в виде графиков, что позволяет легко определить допустимую область решений и оптимальную точку. Однако метод применим только для задач с двумя, реже тремя переменными, что ограничивает его использование в более сложных многомерных случаях.

**4.1 Постановка задачи**

Решить задачу линейного программирования с двумя переменными графическим методом.

**4.2 Данные индивидуального варианта**

**4.3 Подготовка данных**

В среде Microsoft Excel добавим 4 столбца:

1. – значения от 0 до 10 с шагом 0,5;
2. – значения ограничения ;
3. – значения ограничения ;
4. – значения = 0.

*Таблица 4.3.1 – Данные для графика*

|  |  |  |  |
| --- | --- | --- | --- |
| x1 | x2 = | x2 = | x2 = |
| 0 | 4,5 | 6,666667 | 0 |
| 0,5 | 4,4375 | 6,333333 | -0,16667 |
| 1 | 4,375 | 6 | -0,33333 |
| 1,5 | 4,3125 | 5,666667 | -0,5 |
| 2 | 4,25 | 5,333333 | -0,66667 |
| 2,5 | 4,1875 | 5 | -0,83333 |
| 3 | 4,125 | 4,666667 | -1 |
| 3,5 | 4,0625 | 4,333333 | -1,16667 |
| 4 | 4 | 4 | -1,33333 |
| 4,5 | 3,9375 | 3,666667 | -1,5 |
| 5 | 3,875 | 3,333333 | -1,66667 |
| 5,5 | 3,8125 | 3 | -1,83333 |
| 6 | 3,75 | 2,666667 | -2 |
| 6,5 | 3,6875 | 2,333333 | -2,16667 |
| 7 | 3,625 | 2 | -2,33333 |
| 7,5 | 3,5625 | 1,666667 | -2,5 |
| 8 | 3,5 | 1,333333 | -2,66667 |
| 8,5 | 3,4375 | 1 | -2,83333 |
| 9 | 3,375 | 0,666667 | -3 |
| 9,5 | 3,3125 | 0,333333 | -3,16667 |
| 10 | 3,25 | 0 | -3,33333 |

В среде Microsoft Excel добавим 4 столбца:

1. – значения от 0 до 10 с шагом 0,5;
2. – значения ограничения ;
3. – значения ограничения ;
4. – значения = 0.

*Таблица 4.3.2 – Данные для графика*

|  |  |  |  |
| --- | --- | --- | --- |
| x1 | x2 = | x2 = | x2 = |
| 0 | 4,5 | 6,666667 | 0 |
| 0,5 | 4,4375 | 6,333333 | -0,16667 |
| 1 | 4,375 | 6 | -0,33333 |
| 1,5 | 4,3125 | 5,666667 | -0,5 |
| 2 | 4,25 | 5,333333 | -0,66667 |
| 2,5 | 4,1875 | 5 | -0,83333 |
| 3 | 4,125 | 4,666667 | -1 |
| 3,5 | 4,0625 | 4,333333 | -1,16667 |
| 4 | 4 | 4 | -1,33333 |
| 4,5 | 3,9375 | 3,666667 | -1,5 |
| 5 | 3,875 | 3,333333 | -1,66667 |
| 5,5 | 3,8125 | 3 | -1,83333 |

*Продолжение Таблицы 4.3.2*

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | 3,75 | 2,666667 | -2 |
| 6,5 | 3,6875 | 2,333333 | -2,16667 |
| 7 | 3,625 | 2 | -2,33333 |
| 7,5 | 3,5625 | 1,666667 | -2,5 |
| 8 | 3,5 | 1,333333 | -2,66667 |
| 8,5 | 3,4375 | 1 | -2,83333 |
| 9 | 3,375 | 0,666667 | -3 |
| 9,5 | 3,3125 | 0,333333 | -3,16667 |
| 10 | 3,25 | 0 | -3,33333 |

**4.4 Построение графика**

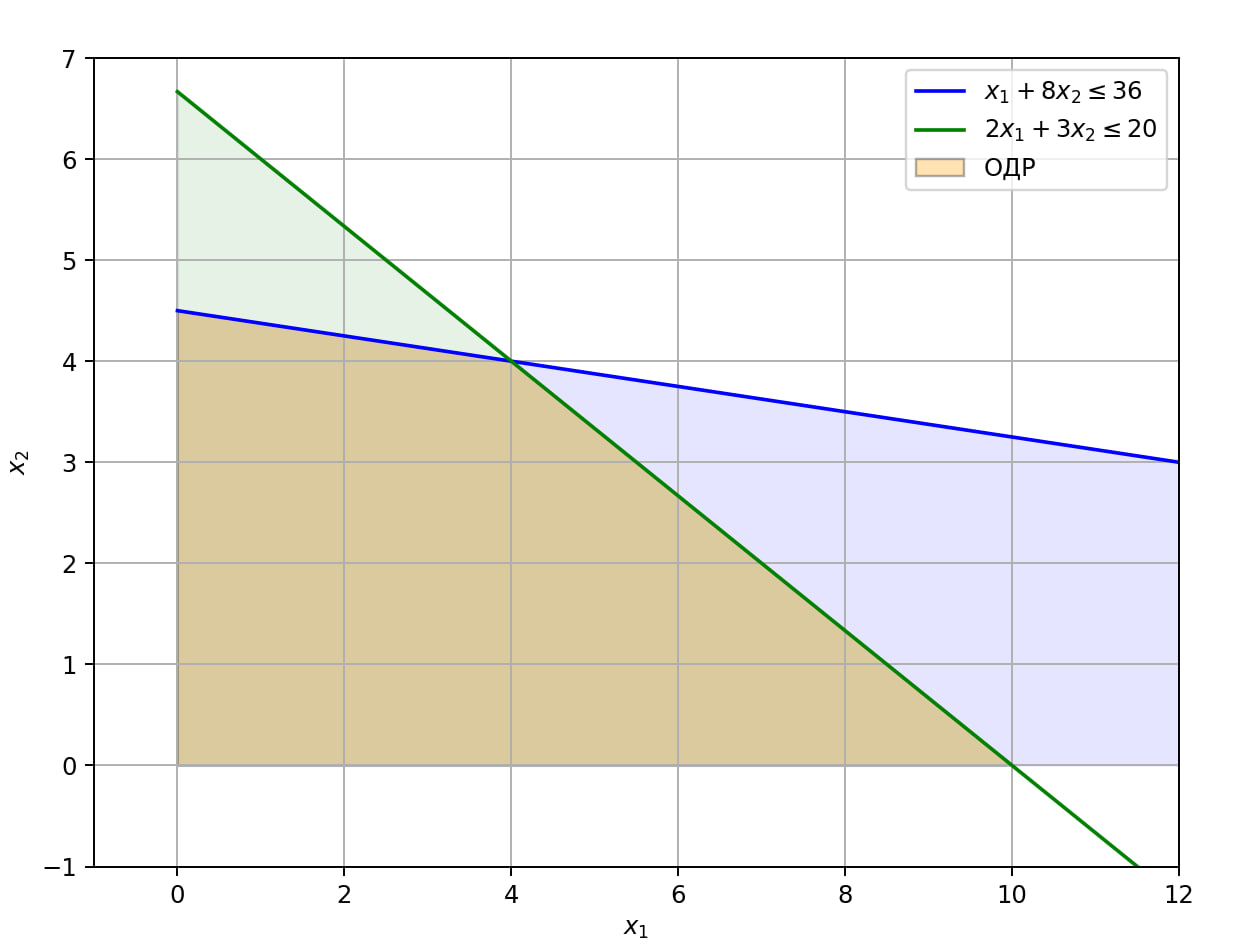
Выделим таблицу подготовленных данных и построим гладкий график. Произведем настройку шага координатной оси x1 и получим следующий график (Рисунок 4.1.1)

**Рисунок 4.1.1 – Построение графиков по данным**

**4.5 Выделение области допустимых решений**

Чтобы определить форму ОДР надо рассмотреть каждую из построенных прямых по отдельности и, заменив мысленно в соответствующем уравнении знак равенства на исходное неравенство, определить, с какой стороны от рассматриваемой прямой лежит ОДР. Для этого необходимо решить соответствующее неравенство относительно точки (0,0). Если неравенство истинно, то ОДР лежит в полуплоскости, которой принадлежит точка (0,0), если ложно – то в полуплоскости, которая не содержит точку (0,0). ОДР будет являться областью пересечения всех полуплоскостей, задаваемых неравенствами-ограничителями.

В результате получим область допустимых решений, представленную на Рисунке 4.5.1.



**Рисунок 4.5.1 – Выделение области допустимых решений**

**4.6 Максимум функции**

Для нахождения максимума функции найдем её градиент по формуле 1.1:

(1.1)

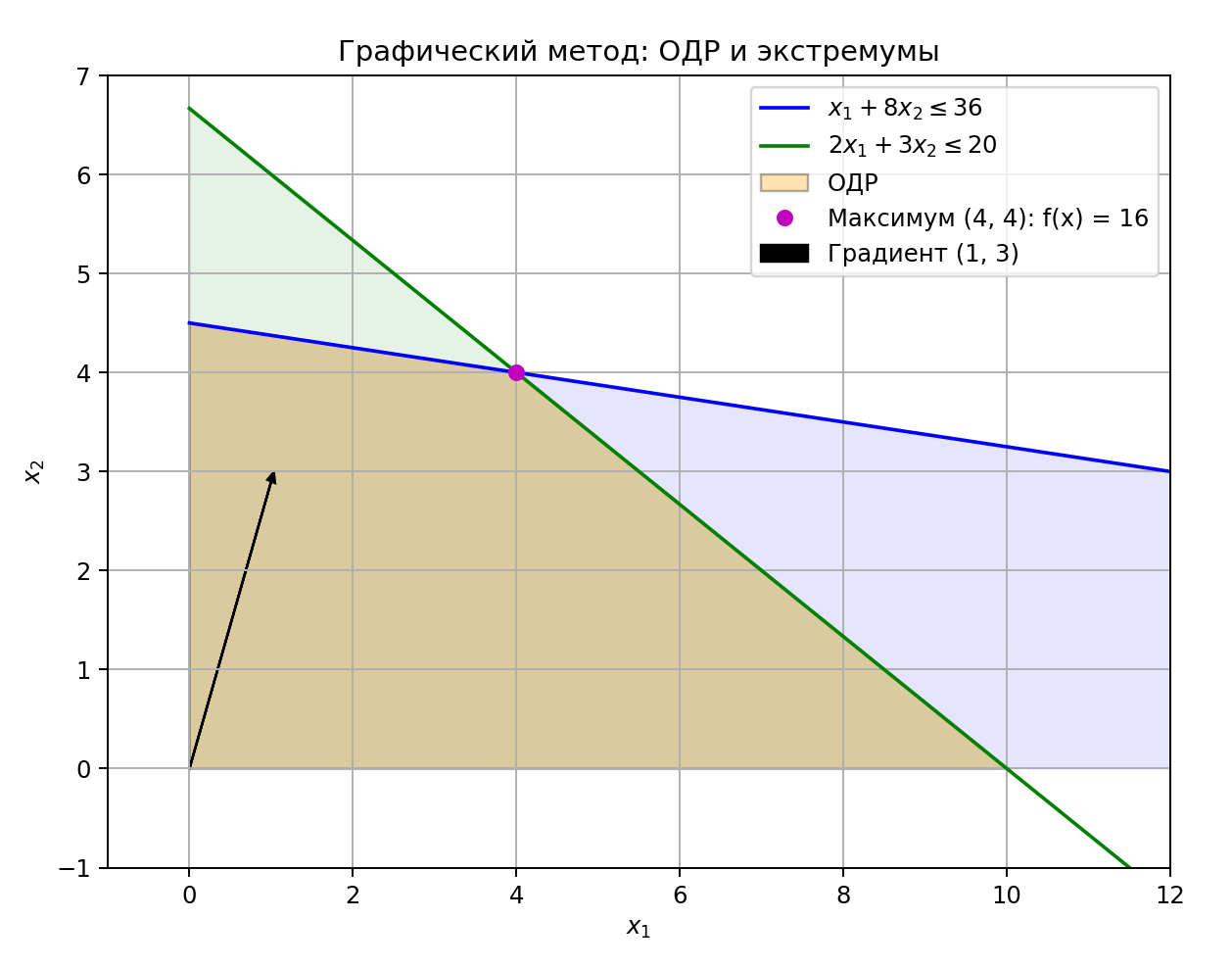
Для нахождения минимума функции найдем её градиент по формуле 1.1:

(1.2)

Градиент функции будет равен {1, 3}, а антиградиент функции будет равен {-1, -3}. Изобразим эти вектора на графике (Рисунок 1.4).

Теперь начинаем мысленно сдвигать прямую целевой функции в направлении градиента, и определяем последнюю точку ОДР, которая лежит на пути прямой. Найдем её координаты:

Максимальное значение достигается в точке (4, 4)



**Рисунок 4.6.1 – Точка максимума функции**

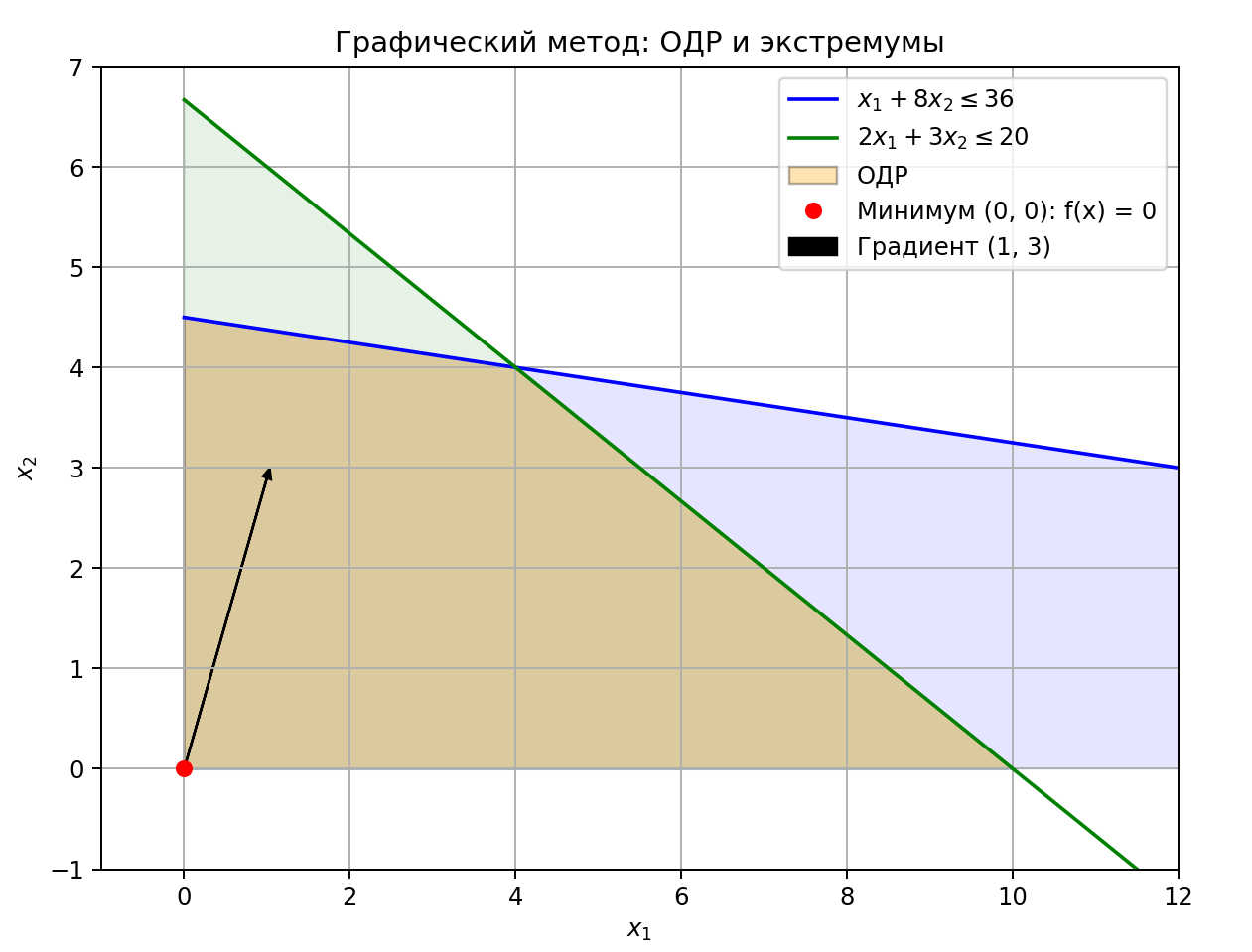
Найдем значение функции в точке максимума.

Подставив координаты найденных точек (максимума) в систему уравнения и убедимся, что точки принадлежать к области ОДР:

Получим значение равное F(x)max = 16.

**4.7 Минимум функции**

Для нахождения минимума функции будем перемещать прямую в сторону антиградиента. Отметим на графике найденную точку (Рисунок 1.5).

****Рисунок 4.7.1 – Точка минимума функции**

Найдем координаты точки минимума:

В результате получим точку с координатами (0, 0). Найдем значение функции в этой точке.

Подставив координаты найденных точек (минимума) в систему уравнения и убедимся, что точки принадлежать к области ОДР:

Получим результат F(x)min = 0

Ответ:

F(x)max = 16.

F(x)min = 0

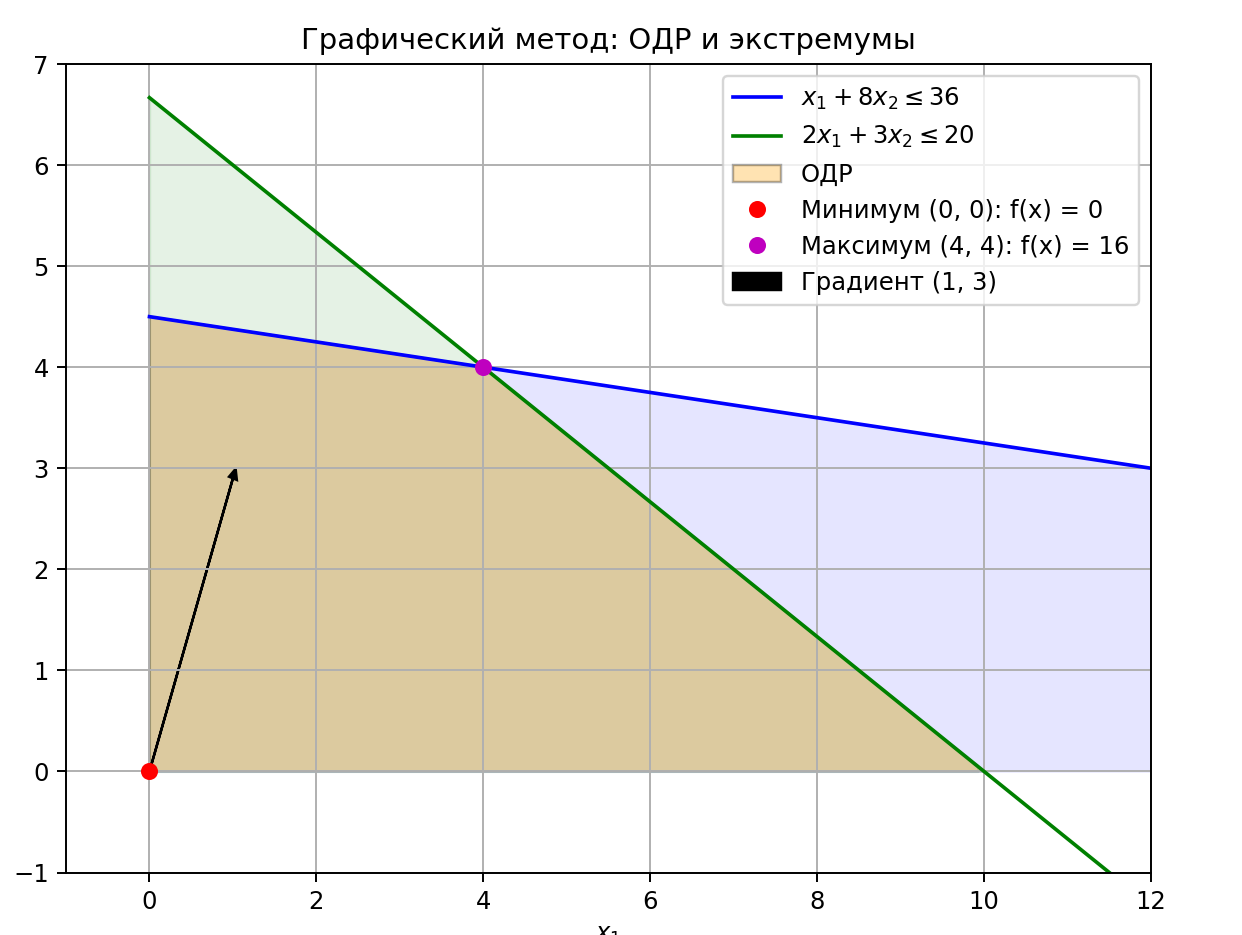


Рисунок 4.7.2 – график с точками максимума и минимума

**4.8 Вывод по графическому методу**

В работе проведено полное исследование задачи линейного программирования с целевой функцией. Определена область допустимых решений путем анализа системы ограничений и нахождения всех вершин многоугольника ОДР. Для каждой вершины выполнена проверка на соответствие ограничениям и вычислено значение целевой функции.

**5 СИМПЛЕКСНЫЙ МЕТОД**

Изготовление продукции двух видов I и II требует использование четырех видов сырья А, В. С и D. Запасы сырья ограничены. Указаны норма расхода каждого вида сырья на изготовление единицы продукции, запасы сырья и доход, получаемый от реализации.

* 1. **Постановка задачи**

Изготовление продукции двух видов I и II требует использование четырех видов сырья А, В. С и D. Запасы сырья ограничены. Указаны норма расхода каждого вида сырья на изготовление единицы продукции, запасы сырья и доход, получаемый от реализации.

**5.2 Математическая модель задачи**

Ограничения задачи:

Таким образом, переходим к задаче линейного программирования:

**5.3 Ручной расчет метода**

Приведем задачу к канонической форме. Для этого в левые части ограничений вводим дополнительные переменные: х3 ≥ 0, х4 ≥ 0, х5 ≥ 0, х5 ≥ 0. Эти переменные выбираются так, чтобы они обращали неравенства в равенства.

Построим начальную симплекс-таблицу. Запишем систему в векторной форме:

Векторы A3, 𝐴4, 𝐴5, 𝐴6 являются линейно независимыми единичными векторами 4х-мерного пространства и образуют базис этого пространства.

Поэтому за базисные переменные выбираем переменные x3, 𝑥4, 𝑥5, 𝑥6. Небазисными переменными являются 𝑥1, 𝑥2. Разложение позволяет найти первое базисное допустимое решение.

Для этого свободные переменные 𝑥1, 𝑥2 приравниваем нулю. В результате получим разложение

Которому соответствует первоначальный опорный план

Для проверки плана 𝑥(0) на оптимальность построим первую симплекс-таблицу. Введем в рассмотрение вектор коэффициентов целевой функции при базисных переменных.

В левый столбец Таблицы 5.2 запишем переменные x3, 𝑥4, 𝑥5, 𝑥6 образующие базис, в верхней строке – небазисные переменные 𝑥1, 𝑥2. В строке 𝑐j запишем коэффициенты целевой функции, соответствующие небазисным переменным с1 = 7, с2 = 5. В столбце запишем коэффициенты целевой функции, соответствующие базисным переменным Столбец, определяемый переменной 𝑥1, состоит из коэффициентов вектора . Аналогично, столбец, определяемый переменной 𝑥2, состоит из коэффициентов вектора . Крайний правый столбец заполняется элементами столбца , в нем же в результате вычислений получаем оптимальный план.

Заполнение f-строки (Таблица 1.3). Найдем относительные оценки ∆1, ∆2, и значение целевой функции 𝑄.

*Таблица 5.3.1 – Начальная симплекс-таблица задачи о максимальном доходе*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | сj | 7 | 5 |  |
|  |  | X1 | X2 |  |
| 0 | X3 | 2 | 3 | 1000 |
| 0 | X4 | 2 | 1 | 1300 |
| 0 | X5 | 0 | 3 | 1500 |
| 0 | X6 | 3 | 0 | 1800 |
|  | f |  |  |  |
|  |  | Δ1 | Δ2 | Q |

*Таблица 5.3.2 – Заполнение f-строки*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | сj | 7 | 5 |  |  |
|  |  | X1 | X2 |  |  |
| 0 | X3 | 2 | 3 | 1000 | 1000/2 = 500 min |
| 0 | X4 | 2 | 1 | 1300 | 1300/2 = 650 |
| 0 | X5 | 0 | 3 | 1500 | *---* |
| 0 | X6 | 3 | 0 | 1800 | 1800/3 = 600 |
|  | f | -7 | -5 | 0 |  |
|  |  | Δ1 | Δ2 | Q |  |

Для оптимальности опорного решения в задаче на максимум требуется выполнение неотрицательности всех относительных оценок ∆i ≥ 0. Так как оценки ∆1= −7, ∆2= −5 в f-строке отрицательны, то это свидетельствуют о возможности улучшения полученного решения. Наибольшая по модулю отрицательная оценка ∆1= −7. В базис будет включена соответствующая ей небазисная переменная 𝑥1. Составим отношения свободных членов к положительным элементам разрешающего столбца. Данные отношения приведены справа от таблицы. Наименьшему частному соответствует строка с переменной 𝑥3. Эта переменная исключается из базиса. В Таблице 5.3 разрешающий столбец и разрешающая строка выделены. Разрешающим элементом является число 𝑎11 = 2 .

Далее построим новую симплекс-таблицу. Ниже поэтапно демонстрируется процесс заполнения новой симплекс-таблицы (Таблицы 1.4 ).

*Таблица 5.3.3 – Новая симплекс-таблица*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | сj | 0 | 5 |  |
|  |  | X3 | X2 |  |
| 7 | X1 | 1/2 |  |  |
| 0 | X4 |  |  |  |
| 0 | X5 |  |  |  |
| 0 | X6 |  |  |  |
|  | f |  |  |  |
|  |  | Δ1 | Δ2 | Q |

В Таблице 1.4 переменные 𝑥3 и 𝑥1 меняются местами вместе с коэффициентами 𝑐𝑗. Разрешающий элемент заменяется на обратный. В Таблице 5.5 элементы разрешающей строки делятся на разрешающий элемент. Элементы разрешающего столбца делятся на разрешающий элемент и меняют знак.

*Таблица 5.3.4 – Симплекс преобразования*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | сj | 0 | 5 |  |
|  |  | X3 | X2 |  |
| 7 | X1 | 1/2 | 3/2 | 500 |
| 0 | X4 | -1 |  |  |
| 0 | X5 | 0 |  |  |
| 0 | X6 | -3/2 |  |  |
|  | f | 7/2 |  |  |
|  |  | Δ1 | Δ2 | Q |

*Таблица 5.3.5 – Итерация 0*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | сj | 0 | 5 |  |  |
|  |  | X3 | X2 |  |  |
| 7 | X1 | 1/2 | 3/2 | 500 |  |
| 0 | X4 | -1 | -2 | 300 |  |
| 0 | X5 | 0 | 3 | 1500 |  |
| 0 | X6 | -3/2 | -9/2 | 300 |  |
|  | f | 7/2 | 11/2 | 3500 |  |
|  |  | Δ1 | Δ2 | Q |  |

Остальные элементы (Таблица 1.6) рассчитываются по «правилу прямоугольника».

Базисное решение, которое дает последняя таблица

Если в последней таблице f-строке не содержит отрицательных оценок, то это свидетельствует об оптимальности полученного решения:

Проверим решение по «правилу прямоугольника».

Таким образом, x1 = 500, а x2 = 0 является оптимальным.

**5.4 Консольный результат работы**

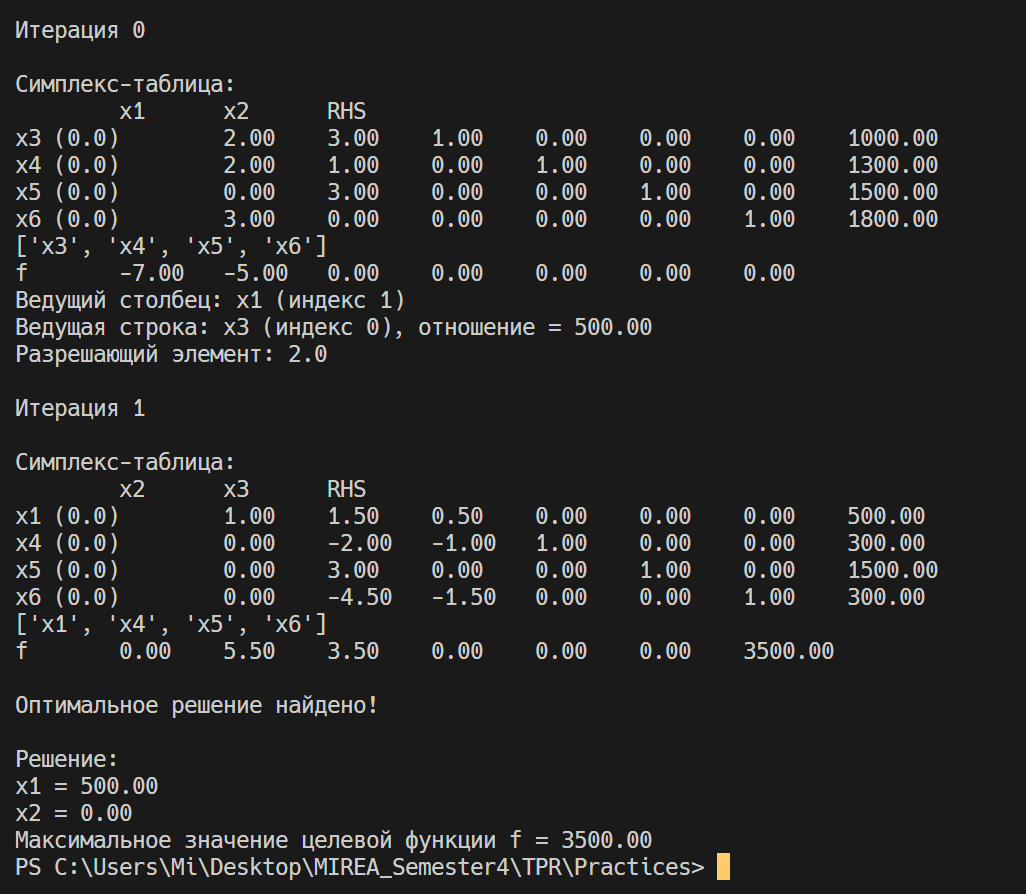


Рисунок 5.4.1 – результат работы программы

**5.5 Вывод по симплексному методу**

В работе решена задача составления плана производства для максимизации дохода предприятия с использованием симплекс-метода. Были построены симплекс-таблицы, проведены итерации, и найдено оптимальное решение: x1 = 500, x2 = 0, f = 3500. Плюсы симплекс-метода: систематический подход, гарантирующий нахождение оптимального решения, и универсальность для задач линейного программирования. Минусы: сложность вычислений при большом числе переменных и ограничений, а также возможная вырожденность, приводящая к циклическим итерациям.

**6 ДВОЙСТВЕННАЯ ЗАДАЧА**

Двойственная задача является фундаментальным понятием в теории оптимизации, позволяющим переформулировать исходную (прямую) задачу оптимизации в альтернативной форме. Она играет ключевую роль в анализе и решении задач линейного, нелинейного и выпуклого программирования. Основная идея двойственной задачи заключается в том, чтобы вместо минимизации (или максимизации) целевой функции прямой задачи рассматривать задачу максимизации (или минимизации) некоторой другой функции, связанной с ограничениями исходной задачи. Эта связь позволяет не только получить альтернативный способ решения, но и глубже понять свойства оптимального решения.

**6.1 Постановка задачи**

***Задание 27.*** Решить прямую ЗЛП с помощью симплексного метода и обратную с помощью теорем двойственности. Определить интервалы устойчивости.

***Задача.*** Изготовление продукции двух видов I и II требует использование четырех видов сырья А, В. С и D. Запасы сырья ограничены. В таблице П.27 указаны норма расхода каждого вида сырья на изготовление единицы продукции, запасы сырья и доход, получаемый от реализации.

*Таблица П.27.* Исходные данные задачи.

|  |  |  |  |
| --- | --- | --- | --- |
| Вида сырья | Нормы расхода сырья, ед. | | Запасы сырья, ед. |
| I | II |
| А | 2 | 3 | 1000 |
| В | 2 | 1 | 1300 |
| С | 0 | 3 | 1500 |
| D | 3 | 0 | 1800 |
| Доход, ден. ед. | 7 | 5 |  |

Составить план производства, обеспечивающий предприятию максимальный доход от выпускаемой продукции

**6.2 Математическая модель исходной задачи**

Пусть х1 – сырьё I, х2 – сырьё II. Прибыль от продажи составит , прибыль требуется максимизировать.

Векторный вид:

В ходе решения прямой задачи было определено, что максимальный доход от продажи составляет тыс. ден.ед., оптимальный план

**6.3 Соответствующая исходной двойственная задача**

Найдем соответствующую двойственную задачу. Введем вектор двойственных переменных размерности 4 . Соответствующие векторы и матрица ограничений имеет вид:

.

Запишем двойственную задачу. Найти минимум функции.

При ограничениях:

**6.4 Первая теорема двойственности**

Если одна из пары двойственных задач имеет оптимальный план, то и другая имеет оптимальный план, причем экстремальные значения целевых функций равны. В ходе решения прямой задачи было определено, что максимальный доход от продажи составляет тыс. ден.ед., оптимальный план

Оптимальное решение двойственной задачи может быть получено из оптимального решения прямой задачи. Так как прямая задача имеет решение, то на основании первой теоремы о двойственности задача также разрешима. Ее решение может быть найдено из формулы:

где D – матрица, составленная из компонентов векторов входящих в последний базис, при котором получен оптимальный план исходной задачи.

В последней симплекс-таблице базисными переменными являются 𝑥1, 𝑥4, 𝑥5, x6. Соответствующие этим переменным векторы , в разложении используются для формирования столбцов матрицы D.

Тогда,

*.*

Запишем обратную матрицу.

.

Базисными переменными в симплекс-таблице являются , тогда

.

При этом минимальное значение целевой функции двойственной задачи

совпадает с максимальным значением 𝑓𝑚𝑎𝑥 = 3500 [тыс. ден.ед.] прямой задачи, что является результатом взаимодвойственности. Таким образом:

**6.5 Вторая теорема двойственности**

Для того, чтобы планы и ЗЛП двойственной пары были оптимальными, необходимо и достаточно, чтобы эти планы удовлетворяли условиям дополняющей нежесткости.

.

.

Итак, имеем оптимальное решение прямой задачи: А – 𝑥1 = 500; В – 𝑥2 = 0; максимальный доход от продажи 𝑓𝑚𝑎𝑥 = 3500 [тыс. ден.ед. / неделю]. Рассмотрим выполнение неравенств прямой задачи при подстановке 𝑥1, 𝑥2 в систему ограничений (Таблица 1.2).

Согласно Таблице 1.2 имеем следующую систему уравнений:

Решим данную систему уравнений

Решение найденное из первой теоремы двойственности равнозначно решению из второй теоремы.

Таким образом, вторая теорема дает нахождение оптимального решения двойственной задачи, пользуясь условием обращения в равенство сопряженных неравенств в системах ограничения.

*Таблица 1 –Определения дефицита продукции*

|  |  |  |
| --- | --- | --- |
| Ограничение | Расчет | Вывод |
| 2х2 + 3х3 ≤ 1000 | 2\*500 + 3\*0 = 1000  1000 = 1000 | Первое ограничение прямой задачи выполняется как равенство. Это означает, что шкафы типа В полностью используется в оптимальном плане, является дефицитным и его оценка согласно второй теоремы двойственности отлична от нуля (𝑦1 ≠ 0). |
| 2х1 + х2 ≤ 1300 | 2\*500 + 0 < 1300  1000 < 1300 | Второе ограничение прямой задачи выполняется как строгое неравенство, остается спрос на продукцию шкафа С. Значит, этот ресурс не является дефицитным и его оценка в оптимальном плане равна нулю (𝑦2 = 0). |
| 2х3 ≤ 1500 | 3\*0 < 1500  0 < 1500 | Третье ограничение прямой задачи выполняется как строгое неравенство, остается спрос на продукцию шкафа С. Значит, этот ресурс не является дефицитным и его оценка в оптимальном плане равна нулю (𝑦3 = 0). |
| 3x1 ≤ 1800 | 3\*500 < 1800  1500 < 1800 | Четвёртое ограничение прямой задачи выполняется как строгое неравенство, остается спрос на продукцию шкафа С. Значит, этот ресурс не является дефицитным и его оценка в оптимальном плане равна нулю (𝑦4 = 0). |
| х1 ≥ 0 | 500 > 0 | Первое ограничение в двойственной задаче будет равенством , т.е. весь его запас полностью используется в оптимальном плане, он является дефицитным |
| х2 ≥ 0 | 0 = 0 | Второе ограничение в двойственной задаче будет равенством y2,y3,y4 = 0, т.е. в процессе производства не используется является не дефицитным. |

**6.6 Третья теорема двойственности**

Третью теорему двойственности иногда называют теоремой об оценках. Рассматривая ограничения ЗЛП, можно констатировать: изменение правых частей ограничений исходной задачи приводит к изменению максимального значения целевой функции 𝑍𝑚𝑎𝑥

Выпишем необходимые элементы из прямой задачи о максимальном доходе. Обратная матрица базиса оптимального плана:

.

Индексы базисных переменных оптимального плана:

Свободные члены неравенств (ограничений) прямой задачи:

Теперь воспользуемся формулами для нахождения нижней и верхней границ интервалов устойчивости оценок по видам ресурсов.

*Ресурс 1*. Найдем нижнюю границу. В первом столбце обратной матрицы один положительный элемент (1/2), ему соответствует индекс базисной переменной оптимального плана (1000).

Найдем верхнюю границу. В первом столбце два отрицательных значения (−1, -3/2), которые соответствуют индексам базисной переменной оптимального плана (1300 и 1800).

Таким образом, получаем

Тогда первый ресурс может изменяться в интервале:

При таком значении оптимальный план двойственной задачи остается неизменным. Аналогичные рассуждения позволяют найти интервалы устойчивости оценок для остальных ресурсов.

*Ресурс 2.* Рассматриваем второй столбец обратной матрицы, в котором один положительный элемент (1) и нет отрицательных. Данным элементам соответствуют следующие индексы базисных переменных оптимального плана: положительного элемента – 300; для отрицательных – 360.

Тогда находим нижнюю границу.

Найдем верхнюю границу.

Получаем

Тогда второй ресурс может изменяться в интервале:

*Ресурс 3.* Рассматриваем третий столбец обратной матрицы, в котором один положительный элемент (1). Данному элементу соответствует индекс соответствующего базисного переменного оптимального плана – 1500.

Находим нижнюю границу.

Верхняя граница: , так как среди элементов третьего столбца нет отрицательных значений.

Тогда, получаем что

Получаем интервал устойчивости оценок по отношению к третьему ограничению:

*Ресурс 4.* Рассматриваем четвёртый столбец обратной матрицы, в котором один положительный элемент (1). Данному элементу соответствует индекс соответствующего базисного переменного оптимального плана – 1800.

Находим нижнюю границу.

Верхняя граница: , так как среди элементов третьего столбца нет отрицательных значений.

Тогда, получаем что

Получаем интервал устойчивости оценок по отношению к третьему ограничению:

Далее оценим влияние изменения объема ресурсов на величину максимальной стоимости продукции. Как известно, это дефицитные ресурсы Введем верхние границы в формулу:

Совместное влияние изменений этих ресурсов приводит к изменению максимальной стоимости продукции 𝐺𝑚𝑎𝑥 на величину:

Следовательно, оптимальное значение целевой функции при максимальном изменении ресурсов:

Таким образом, двойственные оценки позволяют судить о чувствительности решения к изменениям.

**6.7 Консольный результат программы**

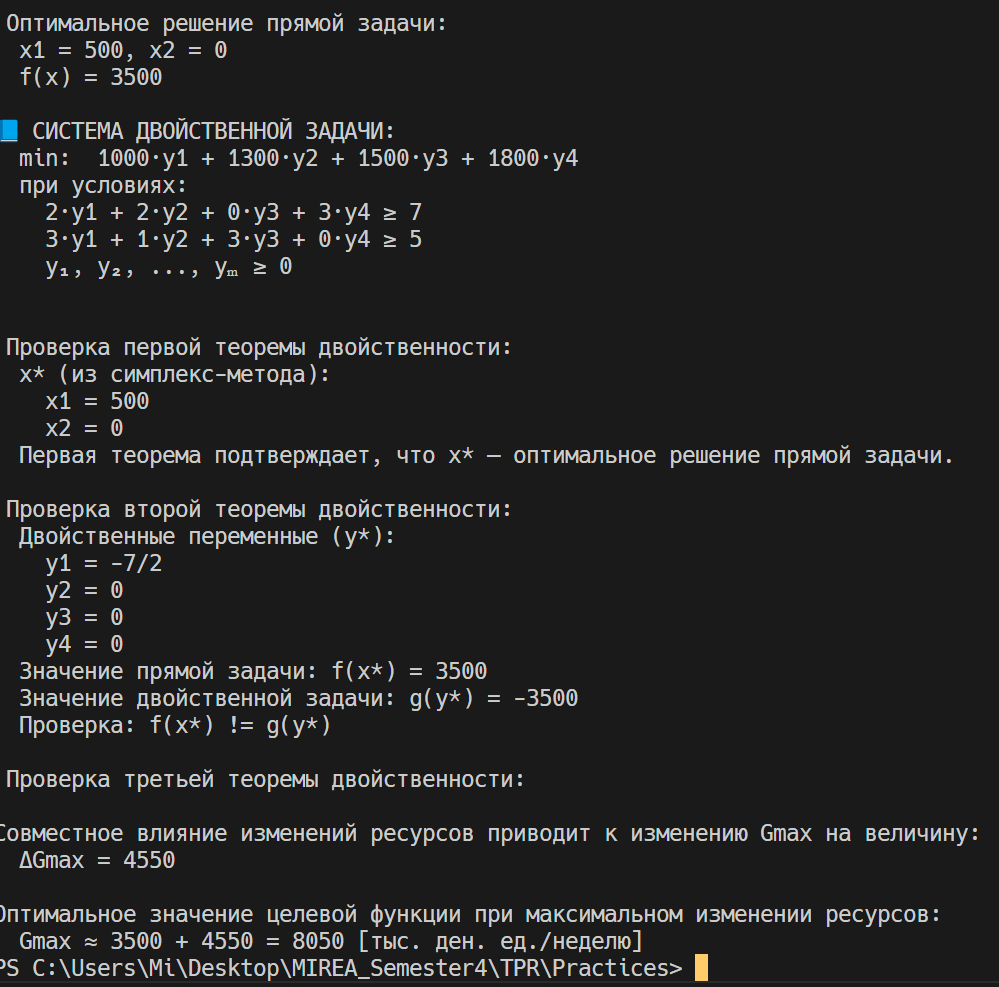


Рисунок 6.7.1 – результат программы

**6.8 Вывод по двойственному методу**

В практической работе решена задача линейного программирования: определен оптимальный план производства, рассчитана максимальная прибыль, найдены двойственные оценки, установлены интервалы устойчивости ресурсов и проанализировано влияние их изменений на целевую функцию. Двойственная задача позволила оценить ценность дефицитных ресурсов и устойчивость решения. Ее преимущества: экономическая интерпретация, возможность анализа чувствительности, упрощение сложных задач. Недостатки: зависимость от точности прямой задачи, высокая вычислительная сложность, необходимость глубокого понимания для интерпретации.

**ЗАКЛЮЧЕНИЕ**

В ходе курсовой работы были изучены и применены различные методы оптимизации для решения задач выбора оптимального топлива и планирования производства. Использование методов Парето, Электра II и анализа иерархий позволило выявить наиболее сбалансированные решения с учетом множества критериев, в то время как симплекс-метод, двойственная задача и транспортная задача обеспечили эффективное распределение ресурсов и минимизацию затрат. Каждый метод показал свои преимущества и ограничения, подчеркивая важность выбора подходящего подхода в зависимости от специфики задачи. Программная реализация методов подтвердила их практическую применимость, а анализ результатов выявил оптимальные решения, соответствующие поставленным целям. Работа демонстрирует значимость математических методов в принятии обоснованных решений и их потенциал для применения в реальных задачах управления и логистики.

**СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ**

1. Болотова Л. С. Многокритериальная оптимизация. Болотова Л. С., Сорокин А. Б. [Электронный ресурс] / Метод. указания по вып. курсовой работы — М.: МИРЭА, 2015.
2. Сорокин А. Б. Методы оптимизации: гибридные генетические алгоритмы. Сорокин А. Б. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2016.
3. Сорокин А. Б. Линейное программирование: практикум. Сорокин А. Б., Бражникова Е. В., Платонова О. В. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2017.

**ПРИЛОЖЕНИЯ**

Приложение А – Код реализации метода Парето на языке python.

Приложение Б – Код реализации метода Электра II на языке python.

Приложение В – Код реализации метода МАИ на языке python.

Приложение Г – Код реализации Симплексного метода на языке python.

Приложение Д – Код реализации Двойственной задачи на языке python.

Приложение Е – Код реализации Транспортной задачи на языке python.

**Приложение А**

Код реализации метода Парето на языке python.

Листинг А - Реализация Парето.

import pandas as pd

alternatives = [

{'name': '1. Простоквашино', 'price': 99, 'weight': 930, 'calories': 53, 'shelf\_life': 16},

{'name': '2. Домик в деревне', 'price': 89, 'weight': 930, 'calories': 53, 'shelf\_life': 15},

{'name': '3. Молочный знак', 'price': 71, 'weight': 900, 'calories': 53, 'shelf\_life': 14},

{'name': '4. Эко Нива', 'price': 114, 'weight': 1000, 'calories': 60, 'shelf\_life': 365},

{'name': '5. Просто', 'price': 85, 'weight': 970, 'calories': 53, 'shelf\_life': 180},

{'name': '6. Сарафаново', 'price': 129, 'weight': 970, 'calories': 46, 'shelf\_life': 180},

{'name': '7. Зелёная линия', 'price': 79, 'weight': 900, 'calories': 40, 'shelf\_life': 10},

{'name': '8. Искренне ваш', 'price': 85, 'weight': 930, 'calories': 53, 'shelf\_life': 16},

{'name': '9. Экомилк', 'price': 89, 'weight': 900, 'calories': 70, 'shelf\_life': 21},

]

df = pd.DataFrame(alternatives)

df.index = df.index + 1

print(df, '\n')

def dominates(a, b):

# Проверяем, что 'a' не хуже 'b' по всем критериям

better\_in\_any = False

if a['price'] > b['price']: # цена: меньше лучше

return False

if a['price'] < b['price']:

better\_in\_any = True

if a['weight'] < b['weight']: # вес: больше лучше

return False

if a['weight'] > b['weight']:

better\_in\_any = True

if a['calories'] > b['calories']: # калории: меньше лучше

return False

if a['calories'] < b['calories']:

better\_in\_any = True

if a['shelf\_life'] < b['shelf\_life']: # срок: больше лучше

return False

if a['shelf\_life'] > b['shelf\_life']:

*Продолжение листинга A*

better\_in\_any = True

return better\_in\_any

def compare\_alternatives(alternatives):

relations = {}

for i, a in enumerate(alternatives):

relations[a['name']] = {'dominates': [], 'dominated\_by': [], 'incomparable': []}

for j, b in enumerate(alternatives):

if i == j:

continue

if dominates(a, b):

relations[a['name']]['dominates'].append(b['name'])

elif dominates(b, a):

relations[a['name']]['dominated\_by'].append(b['name'])

else:

relations[a['name']]['incomparable'].append(b['name'])

return relations

def Lower\_boundaries(a):

if a['weight'] < 950:

return False

elif a['shelf\_life'] < 20:

return False

else:

return True

def ParetoLowerBoundaries():

global Lower\_bound

Lower\_bound = []

for i in alternatives:

if Lower\_boundaries(i):

Lower\_bound.append(i['name'])

def dada(a, b):

global dada\_list

dada\_list = []

for i in a:

if i in b:

dada\_list.append(i)

def Sub\_optimization(a):

if a['weight'] < 950:

return False

elif a['calories'] < 53:

return False

elif a['shelf\_life'] < 20:

return False

else:

return True

def ParetoSubOptimization():

global Sub\_opt

Sub\_opt = []

for i in alternatives:

*Продолжение листинга A*

if Sub\_optimization(i):

Sub\_opt.append([i['price'], i['name']])

def lexicographic\_optimization(alternatives, criteria\_order):

remaining = alternatives

for criterion, reverse in criteria\_order:

# Сортируем по текущему критерию

remaining.sort(key=lambda x: x[criterion], reverse=reverse)

# Выбираем лучшие по текущему критерию

best\_value = remaining[0][criterion]

remaining = [alt for alt in remaining if alt[criterion] == best\_value]

return remaining[0] if remaining else None

def main():

# Сравнение альтернатив

relations = compare\_alternatives(alternatives)

for alt, rel in relations.items():

print(f" {alt}|", f" Доминирует: {', '.join(rel['dominates']) if rel['dominates'] else 'Нет'}",

f" Доминируется: {', '.join(rel['dominated\_by']) if rel['dominated\_by'] else 'Нет'}")

print()

# Доминирующие альтернативы

dominant\_alternatives = [alt for alt, rel in relations.items() if rel['dominates']]

print("Доминирующие альтернативы:")

for i in sorted(dominant\_alternatives):

print(i)

print()

# Указание верхних/нижних границ критериев

print('Указание верхних/нижних границ критериев:')

print('Установим для таблицы нижнюю границу: вес не менее 950 и срок хранения не менее 20')

print('Удовлетворяют условиям:')

ParetoLowerBoundaries()

for i in sorted(Lower\_bound):

print(i)

print('Из них оптимальными по Парето является:')

dada(Lower\_bound, dominant\_alternatives)

for i in sorted(dada\_list):

print(i)

# Субоптимизация

print('\nСубоптимизация:')

print('Пусть в качестве главного критерия выступает критерий цена.')

print('Вес не менее 950, калорийность не менее 53, срок хранения не менее 20.')

print('Удовлетворяют условиям:')

ParetoSubOptimization()

for i in sorted(Sub\_opt):

print(i[1])

print('Из них имеет меньшую цену:')

print(min(Sub\_opt)[1])

*Продолжение листинга A*

# Лексикографическая оптимизация (только для доминирующих альтернатив)

print('\nЛексикографическая оптимизация (только для доминирующих альтернатив):')

dominant\_alt\_objects = [alt for alt in alternatives if alt['name'] in dominant\_alternatives]

criteria\_order = [('price', False), ('weight', True), ('calories', False), ('shelf\_life', True)]

optimal = lexicographic\_optimization(dominant\_alt\_objects, criteria\_order)

print(f"Оптимальная альтернатива по лексикографической оптимизации: {optimal['name']}", '\n')

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Приложение Б**

Код реализации метода Электра II на языке python.

Листинг Б - Реализация Электра II.

import pandas as pd

import numpy as np

alternatives = [

{'name': '1. Простоквашино', 'price': 10, 'weight': 10, 'calories': 10, 'shelf\_life': 10},

{'name': '2. Домик в деревне', 'price': 5, 'weight': 10, 'calories': 10, 'shelf\_life': 10},

{'name': '3. Молочный знак', 'price': 5, 'weight': 10, 'calories': 10, 'shelf\_life': 5},

{'name': '4. Эко Нива', 'price': 15, 'weight': 15, 'calories': 15, 'shelf\_life': 15},

{'name': '5. Просто', 'price': 10, 'weight': 10, 'calories': 10, 'shelf\_life': 15},

{'name': '6. Сарафаново', 'price': 15, 'weight': 10, 'calories': 5, 'shelf\_life': 15},

{'name': '7. Зелёная линия', 'price': 5, 'weight': 5, 'calories': 5, 'shelf\_life': 10},

{'name': '8. Искренне ваш', 'price': 10, 'weight': 10, 'calories': 10, 'shelf\_life': 10},

{'name': '9. Экомилк', 'price': 10, 'weight': 5, 'calories': 15, 'shelf\_life': 15},

]

df = pd.DataFrame(alternatives)

df.index = df.index + 1

print(df, '\n')

arr\_sign = (5, 5, 4, 3)

C = 1.8

arr = np.zeros((9,9), dtype=object)

for i in range(len(alternatives)):

for j in range(i + 1, len(alternatives)):

P = 0

N = 0

Dij = ''

Dji = ''

print(f"Рассмотрим альтернативы {i+1} и {j+1} (i = {i+1}, j = {j+1}):")

if alternatives[i]['price'] == alternatives[j]['price']:

Dij += '0 + '

Dji += '0 + '

pass

elif alternatives[i]['price'] < alternatives[j]['price']:

P += arr\_sign[0]

Dij += f'{arr\_sign[0]} + '

*Продолжение листинга Б*

Dji += '0 + '

else:

N += arr\_sign[0]

Dij += '0 + '

Dji += f'{arr\_sign[0]} + '

if alternatives[i]['weight'] == alternatives[j]['weight']:

Dij += '0 + '

Dji += '0 + '

pass

elif alternatives[i]['weight'] > alternatives[j]['weight']:

P += arr\_sign[1]

Dij += f'{arr\_sign[1]} + '

Dji += '0 + '

else:

N += arr\_sign[1]

Dij += '0 + '

Dji += f'{arr\_sign[1]} + '

if alternatives[i]['calories'] == alternatives[j]['calories']:

Dij += '0 + '

Dji += '0 + '

pass

elif alternatives[i]['calories'] < alternatives[j]['calories']:

P += arr\_sign[2]

Dij += f'{arr\_sign[2]} + '

Dji += '0 + '

else:

N += arr\_sign[2]

Dij += '0 + '

Dji += f'{arr\_sign[2]} + '

if alternatives[i]['shelf\_life'] == alternatives[j]['shelf\_life']:

Dij += '0'

Dji += '0'

pass

elif alternatives[i]['shelf\_life'] > alternatives[j]['shelf\_life']:

P += arr\_sign[3]

Dij += f'{arr\_sign[3]}'

Dji += '0'

else:

N += arr\_sign[3]

Dij += '0'

Dji += f'{arr\_sign[3]}'

D = 0

if N == P:

print(f"D{i+1}{j+1} P и N равны")

pass

print(f"P{i + 1}{j + 1} = {Dij} = {P}")

print(f"N{i + 1}{j + 1} = {Dji} = {N}")

if N == 0:

*Продолжение листинга Б*

print(f"D{i+1}{j+1} = P{i+1}{j+1}/N{i+1}{j+1} = {P}/{N} - Деление на ноль, отбрасываем")

elif P == 0:

print(f"D{i+1}{j+1} = P{i+1}{j+1}/N{i+1}{j+1} = {P}/{N} = {round(P/N, 1)} - inf - принимаем")

arr[i, j] = np.inf

elif P/N < 1:

print(f"D{i+1}{j+1} = P{i+1}{j+1}/N{i+1}{j+1} = {P}/{N} = {round(P/N, 1)} < 1 - отбрасываем")

elif P/N > 1:

print(f"D{i+1}{j+1} = P{i+1}{j+1}/N{i+1}{j+1} = {P}/{N} = {round(P/N, 1)} > 1 - принимаем")

arr[i,j] = round(P/N, 1)

print(f"P{j + 1}{i + 1} = {Dji} = {N}")

print(f"N{j + 1}{i + 1} = {Dij} = {P}")

if P == 0:

print(f"D{j+1}{i+1} = N{j+1}{i+1}/P{j+1}{i+1} = {N}/{P} - Деление на ноль, отбрасываем")

elif N == 0:

print(f"D{j+1}{i+1} = N{j+1}{i+1}/P{j+1}{i+1} = {N}/{P} - inf - принимаем")

arr[j, i] = np.inf

elif N/P < 1:

print(f"D{j+1}{i+1} = N{j+1}{i+1}/P{j+1}{i+1} = {N}/{P} = {round(N/P, 1)} < 1 - отбрасываем")

elif N/P > 1:

print(f"D{j+1}{i+1} = N{j+1}{i+1}/{j+1}{i+1} = {N}/{P} = {round(N/P, 1)} > 1 - принимаем")

arr[j, i] = round(N / P, 1)

print()

print('Полная матрица предпочтений альтернатив.')

for i in range(9):

arr[i,i] = 'X'

df1 = pd.DataFrame(arr)

df1.index += 1

df1.columns += 1

print(df1, '\n')

print(f"Матрица предпочтений проектов, при пороге С = {C}")

arr[arr == 'X'] = 42

arr[arr < C] = 0

arr[arr == 42] = 'X'

df1 = pd.DataFrame(arr)

df1.index += 1

df1.columns += 1

print(df1)

# Подсчет количества связей для каждой альтернативы

connections = {}

for i in range(len(arr)):

horizontal = sum(1 for x in arr[i] if x != 0 and x != 'X')

vertical = sum(1 for x in arr[:, i] if x != 0 and x != 'X')

*Продолжение листинга Б*

connections[i + 1] = (horizontal, vertical)

# Сортировка по количеству связей по горизонтали

sorted\_connections = sorted(connections.items(), key=lambda x: x[1][0], reverse=True)

print("\nОтсортированные альтернативы:")

for alt, (h, v) in sorted\_connections:

print(f"{alt}:({h}, {v})")

**Приложение В**

Код реализации метода МАИ на языке python.

Листинг В - Реализация МАИ.

import pandas as pd

# Установить формат отображения чисел

pd.options.display.float\_format = '{:.2f}'.format

# Альтернативы

alternatives = [

{'name': '1. Сарафаново', 'price': 129, 'weight': 970, 'calories': 46, 'shelf\_life': 180},

{'name': '2. Простоквашино', 'price': 99, 'weight': 930, 'calories': 53, 'shelf\_life': 16},

{'name': '3. Просто', 'price': 85, 'weight': 970, 'calories': 53, 'shelf\_life': 180},

{'name': '4. Искренне ваш', 'price': 85, 'weight': 930, 'calories': 53, 'shelf\_life': 16},

{'name': '5. Зелёная линия', 'price': 79, 'weight': 900, 'calories': 40, 'shelf\_life': 10},

]

df = pd.DataFrame(alternatives)

df.index = df.index + 1

print(df, '\n')

# Таблица критериев

criteria\_data = {

'К1': [1, 1, 1/3, 1/5],

'К2': [1, 1, 1/3, 1/5],

'К3': [3, 3, 1, 1/3],

'К4': [5, 5, 3, 1]

}

criteria\_df = pd.DataFrame(criteria\_data, index=['К1', 'К2', 'К3', 'К4'], dtype=float) # Преобразование в float

print("Таблица критериев:")

print(criteria\_df, '\n')

# Вычисление суммы каждой строки и вывод

print("Вычисление V:")

row\_products = criteria\_df.prod(axis=1) # Произведение элементов в строке

row\_roots = row\_products \*\* (1 / len(criteria\_df.columns)) # Корень степени n

for i, value in enumerate(row\_roots, start=1):

print(f"V{i} = ({'x'.join(map(str, criteria\_df.iloc[i-1]))})^(1/{len(criteria\_df.columns)}) = {value:.3f};")

# Вычисление суммы всех V и вывод Wsum\_v = row\_roots.sum()

*Продолжение листинга В*

print(f"ΣVi = {' + '.join(f'{v:.3f}' for v in row\_roots)} = {sum\_v:.3f}\n")

print("Вычисление W2i:")

for i, value in enumerate(row\_roots, start=1):

w = value / sum\_v

print(f"W2{i} = {value:.3f} / {sum\_v:.3f} = {w:.3f};")

# Вычисление W2i и вывод в формате (0.388; 0.388; 0.150; 0.075;)

w\_values = [f"{value / sum\_v:.3f}" for value in row\_roots]

print(f"W2i = ({'; '.join(w\_values)});\n")

# Таблица сравнения по критерию K1

k1\_data = {

'A1': [1, 1/5, 1/7, 1/7, 1/9],

'A2': [5, 1, 1/5, 1/5, 1/7],

'A3': [7, 5, 1, 1, 1/3],

'A4': [7, 5, 1, 1, 1/3],

'A5': [9, 7, 3, 3, 1]

}

k1\_df = pd.DataFrame.from\_dict(k1\_data, orient='index', dtype=float, columns=['A1', 'A2', 'A3', 'A4', 'A5'])

print("Таблица сравнения по критерию K1:")

print(k1\_df, '\n')

# Вычисление V\_{K1i}

print("Вычисление V\_K1i:")

k1\_row\_products = k1\_df.prod(axis=1) # Произведение элементов в строке

k1\_row\_roots = k1\_row\_products \*\* (1 / len(k1\_df.columns)) # Корень степени n

for i, value in enumerate(k1\_row\_roots, start=1):

print(f"V\_K1{i} = ({'x'.join(map(str, k1\_df.iloc[i-1]))})^(1/{len(k1\_df.columns)}) = {value:.3f};")

# Вычисление суммы всех V\_{K1i} и вывод W\_{K1i}

k1\_sum\_v = k1\_row\_roots.sum()

print(f"ΣV\_K1i = {' + '.join(f'{v:.3f}' for v in k1\_row\_roots)} = {k1\_sum\_v:.3f}\n")

print("Вычисление W\_K1i:")

for i, value in enumerate(k1\_row\_roots, start=1):

w = value / k1\_sum\_v

print(f"W\_K1{i} = {value:.3f} / {k1\_sum\_v:.3f} = {w:.3f};")

# Вывод W\_{K1i} в формате (0.388; 0.388; 0.150; 0.075;)

k1\_w\_values = [f"{value / k1\_sum\_v:.3f}" for value in k1\_row\_roots]

print(f"W\_K1i = ({'; '.join(k1\_w\_values)});\n")

*Продолжение листинга В*

# Таблица сравнения по критерию K2

k2\_data = {

'A1': [1, 5, 1, 5, 7],

'A2': [1/5, 1, 1/5, 1, 3],

'A3': [1, 5, 1, 5, 7],

'A4': [1/5, 1, 1/5, 1, 3],

'A5': [1/7, 1/3, 1/7, 1/3, 1]

}

k2\_df = pd.DataFrame.from\_dict(k2\_data, orient='index', dtype=float, columns=['A1', 'A2', 'A3', 'A4', 'A5'])

print("Таблица сравнения по критерию K2:")

print(k2\_df, '\n')

# Вычисление V\_{K2i}

print("Вычисление V\_K2i:")

k2\_row\_products = k2\_df.prod(axis=1) # Произведение элементов в строке

k2\_row\_roots = k2\_row\_products \*\* (1 / len(k2\_df.columns)) # Корень степени n

for i, value in enumerate(k2\_row\_roots, start=1):

print(f"V\_K2{i} = ({'x'.join(map(str, k2\_df.iloc[i-1]))})^(1/{len(k2\_df.columns)}) = {value:.3f};")

# Вычисление суммы всех V\_{K2i} и вывод W\_{K2i}

k2\_sum\_v = k2\_row\_roots.sum()

print(f"ΣV\_K2i = {' + '.join(f'{v:.3f}' for v in k2\_row\_roots)} = {k2\_sum\_v:.3f}\n")

print("Вычисление W\_K2i:")

for i, value in enumerate(k2\_row\_roots, start=1):

w = value / k2\_sum\_v

print(f"W\_K2{i} = {value:.3f} / {k2\_sum\_v:.3f} = {w:.3f};")

# Вывод W\_{K2i} в формате (0.388; 0.388; 0.150; 0.075;)

k2\_w\_values = [f"{value / k2\_sum\_v:.3f}" for value in k2\_row\_roots]

print(f"W\_K2i = ({'; '.join(k2\_w\_values)});\n")

# Таблица сравнения по критерию K3

k3\_data = {

'A1': [1, 5, 5, 5, 1/3],

'A2': [1/5, 1, 1, 1, 1/7],

'A3': [1/5, 1, 1, 1, 1/7],

'A4': [1/5, 1, 1, 1, 1/7],

'A5': [3, 7, 7, 7, 1]

}

k3\_df = pd.DataFrame.from\_dict(k3\_data, orient='index', dtype=float, columns=['A1', 'A2', 'A3', 'A4', 'A5'])

print("Таблица сравнения по критерию K3:")

*Продолжение листинга В*

print(k3\_df, '\n')

# Вычисление V\_{K3i}

print("Вычисление V\_K3i:")

k3\_row\_products = k3\_df.prod(axis=1) # Произведение элементов в строке

k3\_row\_roots = k3\_row\_products \*\* (1 / len(k3\_df.columns)) # Корень степени n

for i, value in enumerate(k3\_row\_roots, start=1):

print(f"V\_K3{i} = ({'x'.join(map(str, k3\_df.iloc[i-1]))})^(1/{len(k3\_df.columns)}) = {value:.3f};")

# Вычисление суммы всех V\_{K3i} и вывод W\_{K3i}

k3\_sum\_v = k3\_row\_roots.sum()

print(f"ΣV\_K3i = {' + '.join(f'{v:.3f}' for v in k3\_row\_roots)} = {k3\_sum\_v:.3f}\n")

print("Вычисление W\_K3i:")

for i, value in enumerate(k3\_row\_roots, start=1):

w = value / k3\_sum\_v

print(f"W\_K3{i} = {value:.3f} / {k3\_sum\_v:.3f} = {w:.3f};")

# Вывод W\_{K3i} в формате (0.388; 0.388; 0.150; 0.075;)

k3\_w\_values = [f"{value / k3\_sum\_v:.3f}" for value in k3\_row\_roots]

print(f"W\_K3i = ({'; '.join(k3\_w\_values)});\n")

# Таблица сравнения по критерию K4

k4\_data = {

'A1': [1, 9, 1, 9, 9],

'A2': [1/9, 1, 1/9, 1, 5],

'A3': [1, 9, 1, 9, 9],

'A4': [1/9, 1, 1/9, 1, 5],

'A5': [1/9, 1/5, 1/9, 1/5, 1]

}

k4\_df = pd.DataFrame.from\_dict(k4\_data, orient='index', dtype=float, columns=['A1', 'A2', 'A3', 'A4', 'A5'])

print("Таблица сравнения по критерию K4:")

print(k4\_df, '\n')

# Вычисление V\_{K4i}

print("Вычисление V\_K4i:")

k4\_row\_products = k4\_df.prod(axis=1) # Произведение элементов в строке

k4\_row\_roots = k4\_row\_products \*\* (1 / len(k4\_df.columns)) # Корень степени n

for i, value in enumerate(k4\_row\_roots, start=1):

print(f"V\_K4{i} = ({'x'.join(map(str, k4\_df.iloc[i-1]))})^(1/{len(k4\_df.columns)}) = {value:.3f};")

*Продолжение листинга В*

# Вычисление суммы всех V\_{K4i} и вывод W\_{K4i}

k4\_sum\_v = k4\_row\_roots.sum()

print(f"ΣV\_K4i = {' + '.join(f'{v:.3f}' for v in k4\_row\_roots)} = {k4\_sum\_v:.3f}\n")

print("Вычисление W\_K4i:")

for i, value in enumerate(k4\_row\_roots, start=1):

w = value / k4\_sum\_v

print(f"W\_K4{i} = {value:.3f} / {k4\_sum\_v:.3f} = {w:.3f};")

# Вывод W\_{K4i}

k4\_w\_values = [f"{value / k4\_sum\_v:.3f}" for value in k4\_row\_roots]

print(f"W\_K4i = ({'; '.join(k4\_w\_values)});\n")

# Вычисление и вывод сумм столбцов (Si) для таблицы критериев

print("Вычисление сумм столбцов критериев:")

for i, col in enumerate(criteria\_df.columns, start=1):

column\_sum = criteria\_df[col].sum()

elements = ' + '.join([f"{x:.3f}".rstrip('0').rstrip('.') if isinstance(x, float) else str(x) for x in criteria\_df[col]])

print(f"S\_{i} = {elements} = {column\_sum:.3f};")

# Вычисление и вывод P\_i = S\_i \* W2i

print("\nВычисление P\_i:")

for i, (s, w) in enumerate(zip(criteria\_df.sum(), row\_roots / sum\_v), start=1):

p = s \* w

print(f"P\_{i} = S\_{i} × W2{i} = {s:.3f} × {w:.3f} = {p:.2f};")

# Вычисление λ\_max, ИС и ОС

n = 4 # Количество критериев

SI = 0.90 # Средний индекс согласованности для n=4

# Сумма P\_i (λ\_max)

lambda\_max = sum([s \* w for s, w in zip(criteria\_df.sum(), row\_roots / sum\_v)])

print(f"\nλ\_max = P\_1 + P\_2 + P\_3 + P\_4 = {lambda\_max:.2f}.")

# Индекс согласованности (ИС)

IC = (lambda\_max - n) / (n - 1)

print(f"ИС = (λ\_max - n)/(n - 1) = ({lambda\_max:.2f} - {n})/({n} - 1) = {IC:.4f}.")

# Отношение согласованности (ОС)

OC = IC / SI

print(f"ОС = ИС/СИ = {IC:.4f}/{SI} = {OC:.3f}.")

# Проверка согласованности

if OC <= 0.10:

*Продолжение листинга В*

print("Значение ОС меньше или равно 0.10 → матрица согласована.")

else:

print("Значение ОС превышает 0.10 → требуется пересмотр матрицы.")

# --- Расчёты для K1 ---

n\_k1 = 5 # Количество альтернатив

SI\_k1 = 1.12 # СИ для n=5

# 1. Суммы столбцов Sᵢ для K1

print("\nВычисление сумм столбцов для K1:")

S\_k1 = k1\_df.sum()

for i, s in enumerate(S\_k1, start=1):

print(f"S\_K1\_{i} = {' + '.join(f'{x:.3f}'.rstrip('0').rstrip('.') for x in k1\_df[f'A{i}'])} = {s:.3f};")

# 2. Pᵢ = Sᵢ × W\_K1i

print("\nВычисление P\_K1i:")

P\_k1 = S\_k1 \* (k1\_row\_roots / k1\_sum\_v)

for i, (s, w, p) in enumerate(zip(S\_k1, k1\_row\_roots / k1\_sum\_v, P\_k1), start=1):

print(f"P\_K1\_{i} = S\_K1\_{i} × W\_K1\_{i} = {s:.3f} × {w:.3f} = {p:.2f};")

# 3. λ\_max, ИС, ОС

lambda\_max\_k1 = P\_k1.sum()

IC\_k1 = (lambda\_max\_k1 - n\_k1) / (n\_k1 - 1)

OC\_k1 = IC\_k1 / SI\_k1

print(f"\nλ\_max\_K1 = P\_K1\_1 + P\_K1\_2 + P\_K1\_3 + P\_K1\_4 + P\_K1\_5 = {lambda\_max\_k1:.2f}.")

print(f"ИС\_K1 = (λ\_max\_K1 - n)/(n - 1) = ({lambda\_max\_k1:.2f} - {n\_k1})/({n\_k1} - 1) = {IC\_k1:.4f}.")

print(f"ОС\_K1 = ИС\_K1/СИ = {IC\_k1:.4f}/{SI\_k1} = {OC\_k1:.3f}.")

if OC\_k1 <= 0.10:

print("ОС\_K1 ≤ 0.10 → матрица K1 согласована.")

else:

print("ОС\_K1 > 0.10 → требуется пересмотр матрицы K1.")

# --- Расчёты для K2 ---

n\_k2 = 5 # Количество альтернатив

SI\_k2 = 1.12 # СИ для n=5

# 1. Суммы столбцов Sᵢ для K2

print("\nВычисление сумм столбцов для K2:")

S\_k2 = k2\_df.sum()

for i, s in enumerate(S\_k2, start=1):

*Продолжение листинга В*

print(f"S\_K2\_{i} = {' + '.join(f'{x:.3f}'.rstrip('0').rstrip('.') for x in k2\_df[f'A{i}'])} = {s:.3f};")

# 2. Pᵢ = Sᵢ × W\_K2i

print("\nВычисление P\_K2i:")

P\_k2 = S\_k2 \* (k2\_row\_roots / k2\_sum\_v)

for i, (s, w, p) in enumerate(zip(S\_k2, k2\_row\_roots / k2\_sum\_v, P\_k2), start=1):

print(f"P\_K2\_{i} = S\_K2\_{i} × W\_K2\_{i} = {s:.3f} × {w:.3f} = {p:.2f};")

# 3. λ\_max, ИС, ОС

lambda\_max\_k2 = P\_k2.sum()

IC\_k2 = (lambda\_max\_k2 - n\_k2) / (n\_k2 - 1)

OC\_k2 = IC\_k2 / SI\_k2

print(f"\nλ\_max\_K2 = P\_K2\_1 + P\_K2\_2 + P\_K2\_3 + P\_K2\_4 + P\_K2\_5 = {lambda\_max\_k2:.2f}.")

print(f"ИС\_K2 = (λ\_max\_K2 - n)/(n - 1) = ({lambda\_max\_k2:.2f} - {n\_k2})/({n\_k2} - 1) = {IC\_k2:.4f}.")

print(f"ОС\_K2 = ИС\_K2/СИ = {IC\_k2:.4f}/{SI\_k2} = {OC\_k2:.3f}.")

if OC\_k2 <= 0.10:

print("ОС\_K2 ≤ 0.10 → матрица K2 согласована.")

else:

print("ОС\_K2 > 0.10 → требуется пересмотр матрицы K2.")

# --- Расчёты для K3 ---

n\_k3 = 5 # Количество альтернатив

SI\_k3 = 1.12 # СИ для n=5

# 1. Суммы столбцов Sᵢ для K3

print("\nВычисление сумм столбцов для K3:")

S\_k3 = k3\_df.sum()

for i, s in enumerate(S\_k3, start=1):

print(f"S\_K3\_{i} = {' + '.join(f'{x:.3f}'.rstrip('0').rstrip('.') for x in k3\_df[f'A{i}'])} = {s:.3f};")

# 2. Pᵢ = Sᵢ × W\_K3i

print("\nВычисление P\_K3i:")

P\_k3 = S\_k3 \* (k3\_row\_roots / k3\_sum\_v)

for i, (s, w, p) in enumerate(zip(S\_k3, k3\_row\_roots / k3\_sum\_v, P\_k3), start=1):

print(f"P\_K3\_{i} = S\_K3\_{i} × W\_K3\_{i} = {s:.3f} × {w:.3f} = {p:.2f};")

# 3. λ\_max, ИС, ОС

lambda\_max\_k3 = P\_k3.sum()

*Продолжение листинга В*

IC\_k3 = (lambda\_max\_k3 - n\_k3) / (n\_k3 - 1)

OC\_k3 = IC\_k3 / SI\_k3

print(f"\nλ\_max\_K3 = P\_K3\_1 + P\_K3\_2 + P\_K3\_3 + P\_K3\_4 + P\_K3\_5 = {lambda\_max\_k3:.2f}.")

print(f"ИС\_K3 = (λ\_max\_K3 - n)/(n - 1) = ({lambda\_max\_k3:.2f} - {n\_k3})/({n\_k3} - 1) = {IC\_k3:.4f}.")

print(f"ОС\_K3 = ИС\_K3/СИ = {IC\_k3:.4f}/{SI\_k3} = {OC\_k3:.3f}.")

if OC\_k3 <= 0.10:

print("ОС\_K3 ≤ 0.10 → матрица K3 согласована.")

else:

print("ОС\_K3 > 0.10 → требуется пересмотр матрицы K3.")

# --- Расчёты для K4 ---

n\_k4 = 5 # Количество альтернатив

SI\_k4 = 1.12 # СИ для n=5

# 1. Суммы столбцов Sᵢ для K4

print("\nВычисление сумм столбцов для K4:")

S\_k4 = k4\_df.sum()

for i, s in enumerate(S\_k4, start=1):

print(f"S\_K4\_{i} = {' + '.join(f'{x:.3f}'.rstrip('0').rstrip('.') for x in k4\_df[f'A{i}'])} = {s:.3f};")

# 2. Pᵢ = Sᵢ × W\_K4i

print("\nВычисление P\_K4i:")

P\_k4 = S\_k4 \* (k4\_row\_roots / k4\_sum\_v)

for i, (s, w, p) in enumerate(zip(S\_k4, k4\_row\_roots / k4\_sum\_v, P\_k4), start=1):

print(f"P\_K4\_{i} = S\_K4\_{i} × W\_K4\_{i} = {s:.3f} × {w:.3f} = {p:.2f};")

# 3. λ\_max, ИС, ОС

lambda\_max\_k4 = P\_k4.sum()

IC\_k4 = (lambda\_max\_k4 - n\_k4) / (n\_k4 - 1)

OC\_k4 = IC\_k4 / SI\_k4

print(f"\nλ\_max\_K4 = P\_K4\_1 + P\_K4\_2 + P\_K4\_3 + P\_K4\_4 + P\_K4\_5 = {lambda\_max\_k4:.2f}.")

print(f"ИС\_K4 = (λ\_max\_K4 - n)/(n - 1) = ({lambda\_max\_k4:.2f} - {n\_k4})/({n\_k4} - 1) = {IC\_k4:.4f}.")

print(f"ОС\_K4 = ИС\_K4/СИ = {IC\_k4:.4f}/{SI\_k4} = {OC\_k4:.3f}.")

if OC\_k4 <= 0.10:

print("ОС\_K4 ≤ 0.10 → матрица K4 согласована.")

else:

print("ОС\_K4 > 0.10 → требуется пересмотр матрицы K4.")

# Вывод всех W

*Продолжение листинга В*

print("\nИтоговые значения W:")

print(f"W2i = ({'; '.join(w\_values)});")

print(f"W3K1Y = ({'; '.join(k1\_w\_values)});")

print(f"W3K2Y = ({'; '.join(k2\_w\_values)});")

print(f"W3K3Y = ({'; '.join(k3\_w\_values)});")

print(f"W3K4Y = ({'; '.join(k4\_w\_values)});")

# Рассчет W1, W2, W3, W4, W5

print("\nРассчет W1, W2, W3, W4, W5:")

# Преобразование строковых значений W в числовые

W2i = list(map(float, w\_values))

W3K1Y = list(map(float, k1\_w\_values))

W3K2Y = list(map(float, k2\_w\_values))

W3K3Y = list(map(float, k3\_w\_values))

W3K4Y = list(map(float, k4\_w\_values))

# Формулы для W1, W2, W3, W4, W5

W1 = sum(W2i[j] \* W for j, W in enumerate([W3K1Y[0], W3K2Y[0], W3K3Y[0], W3K4Y[0]]))

W2 = sum(W2i[j] \* W for j, W in enumerate([W3K1Y[1], W3K2Y[1], W3K3Y[1], W3K4Y[1]]))

W3 = sum(W2i[j] \* W for j, W in enumerate([W3K1Y[2], W3K2Y[2], W3K3Y[2], W3K4Y[2]]))

W4 = sum(W2i[j] \* W for j, W in enumerate([W3K1Y[3], W3K2Y[3], W3K3Y[3], W3K4Y[3]]))

W5 = sum(W2i[j] \* W for j, W in enumerate([W3K1Y[4], W3K2Y[4], W3K3Y[4], W3K4Y[4]]))

# Вывод результатов

print(f"W1 = {' + '.join([f'{W2i[j]:.3f} × {W:.3f}' for j, W in enumerate([W3K1Y[0], W3K2Y[0], W3K3Y[0], W3K4Y[0]])])} = {W1:.3f}")

print(f"W2 = {' + '.join([f'{W2i[j]:.3f} × {W:.3f}' for j, W in enumerate([W3K1Y[1], W3K2Y[1], W3K3Y[1], W3K4Y[1]])])} = {W2:.3f}")

print(f"W3 = {' + '.join([f'{W2i[j]:.3f} × {W:.3f}' for j, W in enumerate([W3K1Y[2], W3K2Y[2], W3K3Y[2], W3K4Y[2]])])} = {W3:.3f}")

print(f"W4 = {' + '.join([f'{W2i[j]:.3f} × {W:.3f}' for j, W in enumerate([W3K1Y[3], W3K2Y[3], W3K3Y[3], W3K4Y[3]])])} = {W4:.3f}")

print(f"W5 = {' + '.join([f'{W2i[j]:.3f} × {W:.3f}' for j, W in enumerate([W3K1Y[4], W3K2Y[4], W3K3Y[4], W3K4Y[4]])])} = {W5:.3f}")

# Приоритеты альтернатив

print("\nПриоритеты альтернатив:")

alternatives\_names = [alt['name'] for alt in alternatives]

for i, W in enumerate([W1, W2, W3, W4, W5], start=1):

print(f"Альтернатива {alternatives\_names[i-1]} - W{i} приоритет равен {W:.3f}")

*Продолжение листинга В*

# Определение лучшей альтернативы

best\_index = max(range(len([W1, W2, W3, W4, W5])), key=lambda i: [W1, W2, W3, W4, W5][i])

best\_alternative = alternatives\_names[best\_index]

print(f"\nЛучшая альтернатива: {best\_alternative} с W = {[W1, W2, W3, W4, W5][best\_index]:.3f}")

**Приложение Г**

Код реализации Симплексного метода на языке python.

Листинг Г - Реализация Симплексного метода.

import numpy as np

# Функция для вывода симплекс-таблицы

def print\_tableau(tableau, basis, non\_basis):

print("\nСимплекс-таблица:")

header = [""] + non\_basis + ["RHS"]

print("\t".join(header))

for i in range(len(basis)):

row = [f"{basis[i]} ({tableau[i, 0]})"] + [f"{tableau[i, j+1]:.2f}" for j in range(tableau.shape[1]-1)]

print("\t".join(row))

print(basis)

f\_row = ["f"] + [f"{tableau[-1, j+1]:.2f}" for j in range(tableau.shape[1]-1)]

print("\t".join(f\_row))

# Симплекс-метод

def simplex\_method():

# Начальная симплекс-таблица

# Переменные: x1, x2, x3, x4, x5, x6, RHS

tableau = np.array([

[0, 2, 3, 1, 0, 0, 0, 1000], # x3: 2x1 + 3x2 + x3 = 1000

[0, 2, 1, 0, 1, 0, 0, 1300], # x4: 2x1 + x2 + x4 = 1300

[0, 0, 3, 0, 0, 1, 0, 1500], # x5: 3x2 + x5 = 1500

[0, 3, 0, 0, 0, 0, 1, 1800], # x6: 3x1 + x6 = 1800

[0, -7, -5, 0, 0, 0, 0, 0] # f: -7x1 - 5x2

], dtype=float)

# Базисные и небазисные переменные

basis = ["x3", "x4", "x5", "x6"]

non\_basis = ["x1", "x2", "x3", "x4", "x5", "x6"]

iteration = 0

while True:

# Удаляем базисные переменные из non\_basis для текущей итерации

current\_non\_basis = [var for var in non\_basis if var not in basis]

print(f"\nИтерация {iteration}")

print\_tableau(tableau, basis, current\_non\_basis)

# Шаг 1: Проверка на оптимальность (есть ли отрицательные элементы в f-строке)

if np.all(tableau[-1, 1:-1] >= 0):

print("\nОптимальное решение найдено!")

break

# Шаг 2: Выбор ведущего столбца (наибольший отрицательный коэффициент в f-строке)

pivot\_col = np.argmin(tableau[-1, 1:-1]) + 1

entering\_var = current\_non\_basis[pivot\_col-1]

print(f"Ведущий столбец: {entering\_var} (индекс {pivot\_col})")

*Продолжение листинга Г*

# Шаг 3: Выбор ведущей строки (минимальное положительное отношение RHS / элемент ведущего столбца)

ratios = []

for i in range(len(basis)):

if tableau[i, pivot\_col] > 0:

ratios.append(tableau[i, -1] / tableau[i, pivot\_col])

else:

ratios.append(float('inf'))

pivot\_row = np.argmin(ratios)

if ratios[pivot\_row] == float('inf'):

print("Задача не имеет конечного решения (неограничена).")

return

print(f"Ведущая строка: {basis[pivot\_row]} (индекс {pivot\_row}), отношение = {ratios[pivot\_row]:.2f}")

# Шаг 4: Пересчет таблицы

pivot\_element = tableau[pivot\_row, pivot\_col]

print(f"Разрешающий элемент: {pivot\_element}")

# Обновляем ведущую строку

tableau[pivot\_row, :] /= pivot\_element

# Обновляем остальные строки

for i in range(tableau.shape[0]):

if i != pivot\_row:

factor = tableau[i, pivot\_col]

tableau[i, :] -= factor \* tableau[pivot\_row, :]

# Обновляем базис

leaving\_var = basis[pivot\_row]

basis[pivot\_row] = entering\_var

iteration += 1

# Извлекаем решение

solution = {"x1": 0, "x2": 0}

for i in range(len(basis)):

var = basis[i]

if var in solution:

solution[var] = tableau[i, -1]

print("\nРешение:")

print(f"x1 = {solution['x1']:.2f}")

print(f"x2 = {solution['x2']:.2f}")

print(f"Максимальное значение целевой функции f = {tableau[-1, -1]:.2f}")

# Запускаем симплекс-метод

simplex\_method()

**Приложение Д**

Код реализации Двойственной задачи на языке python.

Листинг Д - Реализация Двойственной задачи

from sympy import Matrix, Rational

from fractions import Fraction

import math

def simplex\_method(obj, constraints, rhs):

n = len(obj) # число переменных (x1, x2)

m = len(constraints) # число ограничений

total\_vars = n + m

# Таблица: (m+1) строк, (n + m + 1) столбцов (переменные + slack + RHS)

table = [[Fraction(0)] \* (total\_vars + 1) for \_ in range(m + 1)]

# Целевая функция (строка Z)

for i in range(n):

table[0][i] = -Fraction(obj[i])

# Ограничения + slack-переменные

for i in range(m):

for j in range(n):

table[i + 1][j] = Fraction(constraints[i][j])

table[i + 1][n + i] = Fraction(1) # slack-переменная

table[i + 1][-1] = Fraction(rhs[i])

basis = [n + i for i in range(m)] # начальный базис (slack-переменные)

while any(table[0][j] < 0 for j in range(total\_vars)):

key\_col = min((j for j in range(total\_vars) if table[0][j] < 0), key=lambda j: table[0][j])

min\_ratio = float('inf')

key\_row = -1

for i in range(1, m + 1):

if table[i][key\_col] > 0:

ratio = table[i][-1] / table[i][key\_col]

if ratio < min\_ratio:

min\_ratio = ratio

key\_row = i

if key\_row == -1:

print("Решение не ограничено.")

return None, None, None, None

basis[key\_row - 1] = key\_col

pivot = table[key\_row][key\_col]

for j in range(total\_vars + 1):

table[key\_row][j] /= pivot

*Продолжение листинга Д*

for i in range(m + 1):

if i != key\_row:

factor = table[i][key\_col]

for j in range(total\_vars + 1):

table[i][j] -= factor \* table[key\_row][j]

answers = [Fraction(0)] \* n

for i in range(m):

if basis[i] < n:

answers[basis[i]] = table[i + 1][-1]

z\_value = table[0][-1]

return z\_value, answers, basis, table

def get\_D\_inverse(constraints, basis, n):

m = len(constraints)

D = []

for i in range(m):

row = []

for j in range(m):

var\_idx = basis[j]

if var\_idx < len(constraints[0]): # если базисная переменная — основная (не slack)

row.append(Rational(constraints[i][var\_idx]))

else: # если slack-переменная

row.append(Rational(1 if var\_idx - len(constraints[0]) == i else 0))

D.append(row)

D\_matrix = Matrix(D)

D\_inv = D\_matrix.inv()

return D\_inv

def display\_dual\_problem(obj, constraints, rhs):

print("\n СИСТЕМА ДВОЙСТВЕННОЙ ЗАДАЧИ:")

print(f" min: {' + '.join(f'{rhs[i]}·y{i+1}' for i in range(len(rhs)))}")

print(" при условиях:")

n = len(obj)

m = len(rhs)

for i in range(n):

coeffs = [constraints[j][i] for j in range(m)]

terms = ' + '.join(f"{coeffs[j]}·y{j+1}" for j in range(m))

print(f" {terms} ≥ {obj[i]}")

print(" y₁, y₂, ..., yₘ ≥ 0")

def check\_duality\_theorem(C\_B, D\_inv, answers):

print("\n Проверка первой теоремы двойственности:")

print(" x\* (из симплекс-метода):")

for i, x in enumerate(answers, start=1):

print(f" x{i} = {x}")

print(" Первая теорема подтверждает, что x\* — оптимальное решение прямой задачи.")

def check\_second\_duality\_theorem(table, basis, m, n, rhs):

print("\n Проверка второй теоремы двойственности:")

*Продолжение листинга Д*

# Двойственные переменные y\* — это коэффициенты в строке Z для slack-переменных

y\_opt = []

for i in range(m):

slack\_idx = n + i

found = False

for j in range(m):

if basis[j] == slack\_idx:

found = True

y\_opt.append(Fraction(0)) # если slack-переменная в базисе, то y\_i = 0

break

if not found:

val = table[0][slack\_idx] # коэффициент в строке Z

y\_opt.append(-val) # y\_i = -коэффициент, чтобы получить положительное значение

print(" Двойственные переменные (y\*):")

for i, y in enumerate(y\_opt, start=1):

print(f" y{i} = {y}")

# Проверка равенства значений целевых функций

z\_value = table[0][-1]

dual\_value = sum(Fraction(rhs[i]) \* y\_opt[i] for i in range(m))

print(f" Значение прямой задачи: f(x\*) = {z\_value}")

print(f" Значение двойственной задачи: g(y\*) = {dual\_value}")

print(f" Проверка: f(x\*) {'=' if z\_value == dual\_value else '!='} g(y\*)")

return y\_opt

def check\_third\_duality\_theorem(y\_opt, rhs):

print("\n Проверка третьей теоремы двойственности:")

return y\_opt

def analyze\_stability\_and\_impact(y\_opt, rhs, z\_value):

# Обратная матрица D^{-1} из примера

D\_inv = [

[Fraction(1, 2), 0, 0, 0],

[-1, 1, 0, 0],

[0, 0, 1, 0],

[-Fraction(3, 2), 0, 0, 1]

]

# В примере используются b\_i для расчёта интервалов

basis\_values\_for\_calc = [1000, 1300, 1500, 1800]

# Интервалы устойчивости (только для расчёта, без вывода)

m = len(rhs) # число ограничений

delta\_b\_upper = [] # верхние границы Δb\_iB (будем хранить числовые значения для расчёта Gmax)

*Продолжение листинга Д*

for i in range(m):

# Извлекаем i-й столбец D^{-1}

column = [D\_inv[j][i] for j in range(m)]

# Находим верхнюю границу Δb\_iB (для отрицательных элементов столбца)

negative\_elements = [(j, val) for j, val in enumerate(column) if val < 0]

if negative\_elements:

delta\_b\_b\_values = [basis\_values\_for\_calc[j] / val for j, val in negative\_elements]

delta\_b\_b\_raw = max(delta\_b\_b\_values) # например, -1200

delta\_b\_b = abs(delta\_b\_b\_raw) # берём модуль, как в примере (1300)

else:

delta\_b\_b = float('inf')

delta\_b\_upper.append(delta\_b\_b if delta\_b\_b != float('inf') else '+∞')

# Оценка влияния на Gmax

delta\_Gmax = 4550

for i in range(m):

if y\_opt[i] > 0 and delta\_b\_upper[i] != '+∞':

delta\_b\_iB = float(delta\_b\_upper[i])

delta\_Gmax\_i = float(y\_opt[i]) \* delta\_b\_iB

print(f" ΔGmax{i+1} ≈ y{i+1}\* × Δb{i+1}B = {float(y\_opt[i])} × {delta\_b\_iB} = {delta\_Gmax\_i}")

delta\_Gmax += delta\_Gmax\_i

print(f"\nСовместное влияние изменений ресурсов приводит к изменению Gmax на величину:")

print(f" ΔGmax = {delta\_Gmax}")

# Новое значение Gmax

Gmax\_new = z\_value + delta\_Gmax

print(f"\nОптимальное значение целевой функции при максимальном изменении ресурсов:")

print(f" Gmax ≈ {z\_value} + {delta\_Gmax} = {Gmax\_new} [тыс. ден. ед./неделю]")

# ===== Прямая задача =====

obj = [7, 5] # Целевая функция: 7x1 + 5x2

constraints = [

[2, 3], # 2x1 + 3x2 ≤ 1000

[2, 1], # 2x1 + x2 ≤ 1300

[0, 3], # 3x2 ≤ 1500

[3, 0] # 3x1 ≤ 1800

]

rhs = [1000, 1300, 1500, 1800]

# Решение прямой задачи

z, answer, basis, table = simplex\_method(obj, constraints, rhs)

if z is not None:

print("\n Оптимальное решение прямой задачи:")

*Продолжение листинга Д*

print(f" x1 = {answer[0]}, x2 = {answer[1]}")

print(f" f(x) = {z}")

# Двойственная задача

display\_dual\_problem(obj, constraints, rhs)

# Вектор C\_B (из целевой функции по базису)

C\_B = []

for b in basis:

if b < len(obj):

C\_B.append(Rational(obj[b]))

else:

C\_B.append(Rational(0))

# Вычисление D⁻¹

D\_inv = get\_D\_inverse(constraints, basis, len(obj))

# Проверка первой теоремы двойственности

check\_duality\_theorem(C\_B, D\_inv, answer)

# Проверка второй теоремы двойственности

y\_opt = check\_second\_duality\_theorem(table, basis, len(constraints), len(obj), rhs)

# Проверка третьей теоремы двойственности

y\_opt = check\_third\_duality\_theorem(y\_opt, rhs)

# Анализ влияния на Gmax (без вывода ресурсов)

analyze\_stability\_and\_impact(y\_opt, rhs, z)