МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Институт информационных технологий и технологического образования
Кафедра информационных технологий и электронного обучения

Основная профессиональная образовательная программа
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль) «Технологии разработки программного обеспечения»
форма обучения – очная

Курсовая работа

по дисциплине «Технологии компьютерного моделирования»

Решение задач линейного программирования
с использованием симплекс метода

Обучающейся 2 курса
Нечаевой Натальи Андреевны

_____

Руководитель:
д.п.н, профессор
_____ Власова Е.З.

«_____» _____ 2024 г.

Санкт-Петербург
2024

ОГЛАВЛЕНИЕ

Введение

В современном мире оптимизация различных процессов является одной из ключевых задач, стоящих перед наукой и промышленностью. Симплекс-метод является одним из наиболее эффективных и широко используемых методов решения задач линейного программирования. В связи с этим, изучение и анализ данного метода представляется крайне актуальным, поскольку он находит свое применение в самых различных областях, таких как экономика, управление, техника, медицина и т.д.

Настоящая работа посвящена методу линейного программирования – симплекс-методу.

Предметом исследования курсовой работы является симплекс-метод как инструмент решения задач линейного программирования.

Цель курсовой работы заключается в детальном изучении и анализе симплекс-метода, рассмотрении его применения для решения конкретных задач линейного программирования, а также программной его реализации с помощью различных библиотек языка программирования Python.

Для достижения цели необходимо решить следующие задачи:

1. Изучить теоретические основы и алгоритмы реализации симплекс-метода, провести анализ научной литературы и интернет-ресурсов по заданной теме.
2. Из изученных данных вывести математическую модель для проведения вычислительного эксперимента.
3. Изучить возможности языка программирования Python при работе с задачами линейного программирования.
4. Изучить эффективность симплекс-метода при решении конкретной задачи.

Основные использованные при написании работы источники литературы представлены в таблице 1.

Таблица 1. Использованные источники литературы

№	Автор	Название	Краткая аннотация
1	Vašek Chvátal	Linear programming	В книге представлены все основные моменты, связанные с линейным программированием, а также даны примеры реализации.
2	Ю. И. Дегтярев	Исследование операций	В книге рассмотрены теория и методы исследования операций, линейного программирования в частности.

Практическая значимость работы: разработанная программа позволяет решать задачи симплекс-методом.

Разработанный программный продукт может быть рекомендован для использования в высших учебных заведениях при изучении основ и принципов линейного программирования в целом и симплекс-метода в частности, а также для ознакомления с возможностями языка программирования Python при решении задач линейного программирования.

Структура курсовой работы включает в себя введение, основную часть, состоящую из теоретической части и практической части, заключение и список литературы.

Теоретическая часть работы посвящена построению математической модели на примере конкретной задачи, а также решение этой задачи вручную при помощи симплекс-таблиц.

Практическая часть работы посвящена обзору двух библиотек языка программирования Python, позволяющих решать задачи линейного программирования.

Содержит 20 страниц, 8 таблиц, 2 рисунка, список источников, содержащий 10 источников и 2 приложения.

Основная часть
Теоретическая часть

Основная задача исследования операций – это поиск таких решений в рамках принятой математической модели, которым отвечают некоторые экстремальные критерии. Линейное программирование относится к области исследования операций, и представляет собой математическую дисциплину, главная задача которой – поиск значений некоторых переменных, доставляющих экстремум функции (1), называемой целевой функцией,

$$z = \sum_{j=1}^{n} c_j x_j \rightarrow \min \text{ (или } max) \quad (1)$$

при условиях (2):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq (\geq) b_1; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq (\geq) b_m. \end{cases} \quad (2)$$

Для решения задачи симплекс-методом, вид (2) должен быть приведен к стандартному виду, что делается с помощью присоединения искусственных (компенсирующих) переменных x_{n+i}:

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n \pm x_{n+1} = b_1; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + \cdots + a_{mn}x_n \pm x_{n+m} = b_m. \end{cases} \quad (3)$$

Знак «плюс» или «минус» выбирается в зависимости от изначального вида неравенств.

Рассмотрим решение задачи симплекс-методом на примере конкретной задачи.

Постановка задачи: имеется некоторый материал в виде металлических листов, которые необходимо раскроить для получения деталей двух типов, причем деталей первого типа должно быть не менее 80 штук, а деталей второго типа – не менее 40 штук. Лист можно раскроить четырьмя известными способами, каждый из которых дает результат, представленный в таблице 2. Требуется так провести операцию изготовления деталей, чтобы общий расход металлических листов оказался минимальным. Каждым способом раскраивается x_j листов, где j – способ раскройки [3].

Таблица 2. Способы раскройки листов

Способ	1	2	3	4
Результат	3 детали типа 1 1 деталь типа 2	2 детали типа 1 6 деталей типа 2	1 деталь типа 1 9 деталей типа 2	0 деталей типа 1 13 деталей типа 2

Целевая функция для конкретной задачи на основании (1) выглядит так:

$$z = x_1 + x_2 + x_3 + x_4 \to min, \tag{4}$$

и будет представлять собой суммарный расход материала.

При подстановке известных значений в (2) получаем следующие неравенства:

$$\begin{cases} 3x_1 + 2x_2 + x_3 + 0x_4 \geq 80; \\ x_1 + 6x_2 + 9x_3 + 13x_4 \geq 40; \\ \forall x_j \geq 0, j \in [1;4]. \end{cases} \tag{5}$$

Таким образом обеспечивается необходимое количество деталей. В рамках этой задачи добавляется условие неотрицательности переменных x_j, так как количество листов не может быть отрицательным.

Для решения задачи симплекс-методом необходимо перейти к стандартному виду с помощью присоединения искусственных (компенсирующих) переменных на основе (3) со знаком «минус», так как в (5) ограничения виде «≥»:

$$\begin{cases} 3x_1 + 2x_2 + x_3 + 0x_4 - x_5 = 80; \\ x_1 + 6x_2 + 9x_3 + 13x_4 - x_6 = 40; \\ \forall x_j \geq 0, j \in [1;6]. \end{cases} \quad (6)$$

Теперь необходимо определить базисные переменные [3]. На данный момент их определить невозможно, так как очевидное базисное решение содержит отрицательные переменные x_5 и x_6.

В каждом из ограничений необходимо выделить базисные переменные, в данном случае их должно быть две. Добавляем следующие искусственные переменные к (6):

$$\begin{cases} 3x_1 + 2x_2 + x_3 - x_5 + x_7 = 80; \\ x_1 + 6x_2 + 9x_3 + 13x_4 - x_6 + x_8 = 40; \\ \forall x_j \geq 0, j \in [1;8]. \end{cases} \quad (7)$$

Добавленные переменные будут являться базисными.

Можно выразить целевую функцию (4), используя добавленные переменные:

$$z = x_1 + x_2 + x_3 + x_4 + 0x_5 + 0x_6 + x_7 + x_8 \rightarrow min. \quad (8)$$

Дальнейшее решение будем проводить при использовании симплекс-таблиц.

Симплекс-таблица представляет собой таблицу, где каждый столбец соответствует одному ограничению или коэффициенту целевой функции, а каждая строка соответствует одной базисной переменной.

Для построения симплекс-таблица выделим массив коэффициентов линейной функции (9) и массив коэффициентов при базисных переменных (10):

$$\{C_i\} = \{1,1,1,1,0,0,1,1\}. \quad (9)$$

$$l_i = \{1,1\}, i \in [1,m]. \quad (10)$$

Также, в таблице будут записаны все коэффициенты при переменных, содержащиеся в целевой функции и ограничениях (a_{ij}).

По последней, «Индексной строке» W_i, будет определяться оптимальность решения.

Так как по условиям задачи функцию необходимо минимизировать, значит, условием оптимальности будет отсутствие положительных значений в строке. Значения в ней будут вычисляться по формуле (11):

$$W_j = \sum_{i=1}^{m} l_i a_{ij} - C_j, \qquad j \in [1,n]. \quad (11)$$

В правом углу таблицы содержится итоговое значение целевой функции с учетом всех элементов таблицы (12):

$$w_{n+1} = \sum_{i=11}^{m} l_{ij} b_i. \quad (12)$$

Симплекс-метод включает в себя несколько итераций, проводимых по одному алгоритму до достижения необходимого результата. В условиях конкретной задачи будет четыре итерации.

Составим первую симплекс-таблицу:

Таблица 3. Итерация 1

B_i	l_i	C_i								b_i
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
		1	1	1	1	0	0	1	1	
x_7	1	3	2	1	0	-1	0	1	0	80
x_8	1	1	6	9	13	0	-1	0	1	40
W_i		3	7	9	12	-1	-1	0	0	120

Наблюдаем, что в последней строке еще присутствуют неотрицательные элементы, что говорит о том, что оптимальное решение еще не достигнуто.

Ищем новую базисную переменную – ей будет та переменная, которая у которой наибольший коэффициент. В зависимости от того, в какой строке (эта строка называется ведущей) содержится этот коэффициент, будет изменен элемент столбца B_i. Все остальные элементы этой строки также будут поделены на максимальный коэффициент (разрешающий элемент), а остальные элементы таблицы (под элементами таблицы здесь и далее подразумеваются только те ячейки, в которых содержатся коэффициенты при ограничениях) изменяются методом прямоугольника относительно разрешающего элемента.

С учетом всех проведенных операций получаем:

Таблица 4. Итерация 2

B_i	l_i	C_i								b_i
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
		1	1	1	1	0	0	1	1	
x_7	1	3	2	1	0	-1	0	1	0	80
x_4	1	$\frac{1}{13}$	$\frac{6}{13}$	$\frac{9}{13}$	1	0	$\frac{-1}{13}$	0	$\frac{1}{13}$	$\frac{40}{13}$
W_i		$3 - \frac{12}{13}$	$2 - \frac{7}{13}$	$10 - \frac{4}{13}$	0	-1	$\frac{-1}{13}$	0	$\frac{1}{13} - 1$	$80 + \frac{40}{13}$

Видно, что в «Индексной строке» все еще есть положительные переменные.

Таблица 5. Итерация 3

B_i	l_i	C_i								b_i
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
		1	1	1	1	0	0	1	1	
x_1	1	1	$\frac{2}{3}$	$\frac{1}{3}$	0	$\frac{-1}{3}$	0	$\frac{1}{3}$	0	$\frac{80}{3}$
x_4	1	0	0,41	0,67	1	0,03	-0,08	-0,03	0,08	1,03
W_i		0	0,08	0	0	-0,3	-0,08	-0,7	0,02	27,7

Все еще имеем положительные значения, поэтому снова повторяем все преобразования, выбрав новую базисную переменную.

Таблица 6. Итерация 4

B_i	l_i	C_i								b_i
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
		1	1	1	1	0	0	1	1	
x_1	1	1	0	-0,75	-1,67	-0,37	0,12	0,38	-1,12	25
x_2	1	0	1	1,6	1	2,4	-0,2	-0,07	0,2	2,5
W_i		0	0	-0,12	-0,19	-0,31	-0,06	-0,69	-0,04	27,5

В «Индексной строке» теперь не содержится положительных элементов, что показывает, что оптимальное решение достигнуто.

По полученным данным можно сделать несколько выводов: для достижения наилучшего результата необходимо кроить 25 листов первым способом и 2,5 листа вторым, при этом ни третий, ни четвертый способ не задействуются, при этом наименьший расход материала составит 27,5 листа. Количество получаемых при этом деталей равно 80 и 40 соответственно.

Практическая часть

Задача данной части заключается в реализации симплекс-метода на языке Python и среды разработки PyCharm с использованием сторонних библиотек для того, чтобы ответить на следующие вопросы:

1. Какими библиотеками можно пользоваться при необходимости реализации задач оптимизации симплекс-методом?
2. Какие библиотеки дают возможность получить более лучшие результаты для реализации симплекс-метода?

Также, в конце этой части представлен алгоритм использования проанализированных библиотек для решения задач симплекс-методом.

Руководствуясь материалами, рассмотренными в Теоретической части, берем математическую модель, которую можно использовать при написании программ:

$$z = x_1 + x_2 + x_3 + x_4 \to min,$$
$$\begin{cases} 3x_1 + 2x_2 + x_3 \geq 80; \\ x_1 + 6x_2 + 9x_3 + 13x_4 \geq 40; \\ \forall x_j \geq 0, j \in [1;4]. \end{cases}$$

Для реализации нам также понадобится предварительно выделить массив коэффициентов линейной функции, матрицу коэффициентов и матрицу свободных членов.

$$\{C_i\} = \{1,1,1,1\}.$$

$$A = \begin{pmatrix} 3 & 2 & 1 & 0 \\ 1 & 6 & 9 & 13 \end{pmatrix}$$

$$B = \begin{pmatrix} 80 \\ 40 \end{pmatrix}$$

В первую очередь рассмотрим использование библиотеки «scipy», предназначенной для проведения инженерных и научных расчетов, и ее

модуль «optimize», содержащий несколько часто используемых алгоритмов оптимизации. Для проведения вычислений потребуется функция «linprog».

Вводимые параметры:

Таблица 7. Переменные

Название	Значение	Назначение
c	[-1, -1, -1, -1]	Коэффициенты целевой функции, умноженные на (-1), так как необходимо искать минимум
A	[[3, 2, 1, 0], [1, 6, 9, 13]]	Ограничения
b	[80, 40]	Правые части ограничений
x1	(0, None)	Ограничение на неотрицательность переменной
x2	(0, None)	Ограничение на неотрицательность переменной
x3	(0, None)	Ограничение на неотрицательность переменной
x4	(0, None)	Ограничение на неотрицательность переменной
res	Вычисляемая функция	Вычисляет все необходимые параметры, после чего их можно вывести в консоль

Вводимая функция выглядит так:

linprog(c, A_ub=A, b_ub=b, bounds=(x1, x2, x3, x4), method='simplex'),

где

- c – коэффициенты целевой функции;
- A_ub – матрица ограничений неравенств;
- b_ub – вектор ограничения неравенств;
- bounds – последовательность пар для каждого элемента x, задающая границы;
- method – указывает, какой метод использовать при расчете.

Код программы представлен в Приложении А.

В результате работы программы в консоль пользователя будет выведена следующая информация:

Рисунок 1. Вывод результатов работы программы

Из рисунка 1 видно, что результат работы программы отличается от результатов ручного просчета. Оптимальное значение целевой функции на ≈0,2 больше, и оптимальное решение совершенно другое. При этом, если провести проверку, то количество получаемых результатов соответствует ожидаемому. Полную документацию к функции с подробным описанием можно получить, нажав на клавиатуре клавишу «Ctrl» и кликнув правой кнопкой мыши по функции «linprog».

Второй рассматриваемой библиотекой будет «ortools», разработанной Google специально для решения задач оптимизации и линейной алгебры, в которой находится модуль «linear_solver». Для решения будет также использоваться внутренняя библиотека «pywraplp».

Данный метод будет длиннее и тяжелее предыдущего, так как он аккуратнее подходит ко всем преобразованиям значений при расчетах значений.

Все наименования, использованные в программе, представлены в таблице 8.

Значения коэффициентов и необходимое действие передается после объявления переменной. Все действия происходят внутри функции.

Таблица 8. Наименования

Название	Значение	Назначение
solve_linear_programming	-	Функция, решающая задачу линейного программирования
solver	pywraplp.Solver.CreateSolver('GLOP')	Экземпляр решателя GLOP, решающего задачу линейного программирования симплекс-методом
x1	solver.NumVar(0, solver.infinity(), 'x1')	Переменная, переданная в решатель, с границами значений
x2	solver.NumVar(0, solver.infinity(), 'x2')	Переменная, переданная в решатель, с границами значений
x3	solver.NumVar(0, solver.infinity(), 'x3')	Переменная, переданная в решатель, с границами значений
x4	solver.NumVar(0, solver.infinity(), 'x4')	Переменная, переданная в решатель, с границами значений
objective	solver.Objective()	Установка целевой функции. В нее будут переданы коэффициенты при переменных, и обозначено, что нужно сделать (min или max)
constraint1	solver.Constraint(80, solver.infinity())	Первое ограничение. Далее сюда будут переданы коэффициенты при каждой из переменных
constraint2	solver.Constraint(40, solver.infinity())	Второе ограничение. Далее сюда будут переданы коэффициенты при каждой из переменных

Значения коэффициентов и необходимое действие передается после объявления переменной. Все действия происходят внутри функции.

Передача переменных производится методом «SetCoefficient» следующим образом:

“название переменной”.SetCoefficient(“имя переменной”, “значение коэффициента”)

Выбор действия:

“название переменной”.SetMinimization() (если необходимо минимизировать функцию, как в нашем случае; в ином используется метод «SetMaximization»)

После объявления всех переменных и передачи всех необходимых значений, производятся вычисления, вызовом метода «Solve»:

“название переменной”.Solve()

После проведения всех вычислений, полученные значения необходимо проверить на оптимальность:

“название переменной”.OPTIMAL

Вычисление и проверку на оптимальность можно сделать как в одной строке, так и разбить на две, как в случае написанной программы в рамках данной работы.

В случае, если достигнут оптимальный результат, результат выводится. В противном случае, если оптимальное решение достигнуто не было, необходимо вывести сообщение об этом пользователю.

Для вывода результата используются функции:

print(“название переменной”.solution_value()) – для вывода оптимальных значений переменных

print('Значение целевой функции: ', objective.Value()) – для вывода оптимального значения целевой функции

Теперь, если вызвать написанную функцию, в консоли пользователя будет выведен результат вычислений:

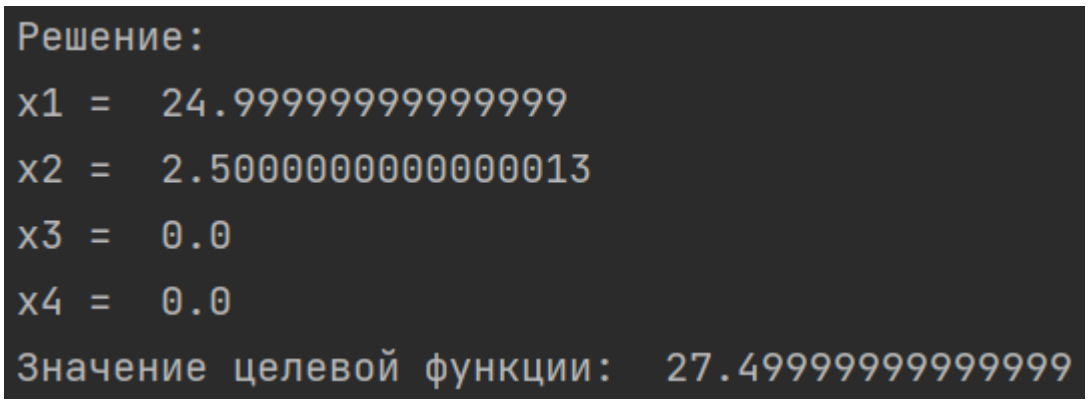

Рисунок 2. Результат работы программы

При использовании данного метода, результат совпадает с тем, что было получено при подсчете с помощью симплекс-таблицы (результат, полученный компьютером, можно смело округлить до целых и, в случае со значением оптимальной функции, до десятых; то, что сама программа считает в таком виде, связано с особенностями вычисления на компьютере).

Полученные данные позволяют сделать вывод, что обе библиотеки, с помощью которых проводились вычисления, подходят для работы с задачами минимизации с использованием симплекс-метода, но предпочтительнее библиотека «ortools», как дающая более оптимальный результат.

Код программы представлен в Приложении Б.

Рассмотрим алгоритм действий, которого следует придерживаться при разработке программы с представленным функционалом (рассматривается работа на Python):

1. Разработать математическую модель, адекватно соответствующую решаемой задаче.
2. Выписать коэффициенты при переменных целевой функции и ограничений.
3. Определить, какие значения могут гипотетически принимать переменные, а также границы ограничений.
4. Импортировать в проект необходимую библиотеку с модулем. Без этого шага программа не будет работать.

5. Определить функцию для вычислений (можно также модернизировать программу так, чтобы пользователь сам вводил все коэффициенты и ограничения).
6. Создать экземпляр решателя GLOP.
7. Создать переменные с их ограничениями.
8. Установить целевую функцию, передать в нее переменные и коэффициенты перед ними.
9. Добавить ограничения, передать в них переменные и коэффициенты перед ними.
10. Выполнить решение задачи, уделив внимание проверке на оптимальность. Если полученное решение соответствует, вывести результат на устройство.

Заключение

В процессе выполнения курсовой работы был проанализирован метод решения задач линейного программирования симплекс-методом, построены математические модели в общем виде и для конкретной задачи. Задача была решена несколькими способами, показан процесс поиска решения с помощью симплекс-таблиц и на языке программирования Python с использованием сторонних библиотек.

Программа реализуется в среде разработки PyCharm, что позволяет любому имеющему на своем персональном компьютере пользователю воспользоваться ею, либо же реализовать собственную, используя алгоритм, данный в Практической части данной работы.

Был сделан вывод, что при необходимости решить задачу линейного программирования симплекс-методом на компьютере, лучше будет воспользоваться библиотекой «ortools», как дающей наиболее точный и оптимальный результат.

Литература

[1] Беников, А. И. Линейное программирование: Учебное пособие / А. И. Беников. – Иркутск: Иркутский университет, 2005. – 147 с. – ISBN 5-9624-0054-2. – Текст: непосредственный

[2] Бурда, А. Г. Исследование операций в экономике : учебное пособие / А. Г. Бурда, Г. П. Бурда. — Санкт-Петербург : Лань, 2022. — 564 с. — ISBN 978-5-8114-3149-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/213143 (дата обращения: 25.05.2024). — Режим доступа: для авториз. Пользователей.

[3] Дегтярев, Ю. И. Исследование операций. – М.: Высшая школа, 1986. – 320 с. – Текст: непосредственный

[4] Кудашов, В. Н. Основы линейного программирования : учебно-методическое пособие / В. Н. Кудашов, Е. Г. Селина. — Санкт-Петербург : НИУ ИТМО, 2020. — 42 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/190804 (дата обращения: 25.05.2024). — Режим доступа: для авториз. Пользователей.

[5] Кудрявцев, К. Я. Методы оптимизации : учебное пособие для вузов / К. Я. Кудрявцев, А. М. Прудников. — 2-е изд. — Москва : Издательство Юрайт, 2024. — 140 с. — (Высшее образование). — ISBN 978-5-534-08523-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/541315 (дата обращения: 27.05.2024).

[6] Корте Б., Фиген Й. Комбинаторная оптимизация. Теория и алгоритмы / Перевод с англ. М. А. Бабенко / Корте Б., Фиген Й. — 2-е изд., стереотип. — М.: МЦНМО, 2015 — 720 с. – Текст: непосредственный

[7] Т.Кормен, Ч. Лейзерсон, Р. Ривест Алгоритмы: построение и анализ / Пер. с англ. под ред. А. Шеня / Т.Кормен, Ч. Лейзерсон, Р. Ривест — 2-е изд., стереотип. — М.: МЦНМО: БИНОМ. Лаборатория знаний, 2004 — 960 с. – Текст: непосредственный

[8] Подробный разбор симплекс-метода // Хабр : сайт. – URL: https://habr.com/ru/articles/474286/ (дата обращения: 27.05.2024)

[9] Bertsimas, Dimitris Introduction to Linear Optimization / Dimitris Bertsimas, John N. Tsitsiklis. – Belmont, Massachusetts : Athena Scientific, 1997. – 587 с. – ISBN 1-886529-19-1. – Текст: непосредственный

[10] Vašek Chvátal Linear programming / Vašek Chvátal — 1st. — New York: W. H. Freeman and Company, 1983 — 500 с. – Текст: непосредственный

Приложение А

```
1  | from scipy.optimize import linprog
2  |
3  | c = [-1, -1, -1, -1]
4  | A = [[3, 2, 1, 0], [1, 6, 9, 13]]
5  | b = [80, 40]
6  | x1 = (0, None)
7  | x2 = (0, None)
8  | x3 = (0, None)
9  | x4 = (0, None)
10| res = linprog(c, A_ub=A, b_ub=b, bounds=(x1, x2, x3, x4), method='simplex')
11|
12| print("Оптимальное решение: ", res.x)
13| print("Оптимальное значение целевой функции: ", -res.fun)
```

Приложение Б

```
2 | from ortools.linear_solver import pywraplp
3 |
4 | def solve_linear_programming():
5 |     solver = pywraplp.Solver.CreateSolver('GLOP')
6 |
7 |     x1 = solver.NumVar(0, solver.infinity(), 'x1')
8 |     x2 = solver.NumVar(0, solver.infinity(), 'x2')
9 |     x3 = solver.NumVar(0, solver.infinity(), 'x3')
10|     x4 = solver.NumVar(0, solver.infinity(), 'x4')
11|
12|     # Устанавливаем целевую функцию
13|     objective = solver.Objective()
14|     objective.SetCoefficient(x1, 1)
15|     objective.SetCoefficient(x2, 1)
16|     objective.SetCoefficient(x3, 1)
17|     objective.SetCoefficient(x4, 1)
18|     objective.SetMinimization()
19|
20|     # Добавляем ограничения
21|     constraint1 = solver.Constraint(80, solver.infinity())
22|     constraint1.SetCoefficient(x1, 3)
23|     constraint1.SetCoefficient(x2, 2)
24|     constraint1.SetCoefficient(x3, 1)
25|     constraint1.SetCoefficient(x4, 0)
26|
27|     constraint2 = solver.Constraint(40, solver.infinity())
28|     constraint2.SetCoefficient(x1, 1)
29|     constraint2.SetCoefficient(x2, 6)
30|     constraint2.SetCoefficient(x3, 9)
31|     constraint2.SetCoefficient(x4, 13)
32|
33|     solver.Solve()
34|
35|     if solver.Solve() == solver.OPTIMAL:
36|         print('Решение:')
37|         print('x1 = ', x1.solution_value())
38|         print('x2 = ', x2.solution_value())
39|         print('x3 = ', x3.solution_value())
40|         print('x4 = ', x4.solution_value())
41|         print('Значение целевой функции: ', objective.Value())
42|     else:
43|         print('Решение не обнаружено.')
44|
45| solve_linear_programming()
```