|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК4 «Программное обеспечение ЭВМ,** |
| **информационные технологии»** | |

**Лабораторная работа №4**

**«Задачи целочисленного линейного программирования»**

**ДИСЦИПЛИНА: «Моделирование»**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-72Б | |  |  | ( | Сафронов Н.С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Никитенко У.В. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

Калуга, 2023

**Цель работы:** сформировать практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для стандартных вычислений при решении задач целочисленного линейного программирования на основе сравнения результатов.

**Постановка задачи**

Найдите оптимальный план задачи целочисленного линейного программирования ( – порядковый номер студента в списке группы), используя

• первый алгоритм Гомори;

• второй алгоритм Гомори ( – произвольное, – целое);

• метод ветвей и границ (решение проиллюстрируйте схемой).

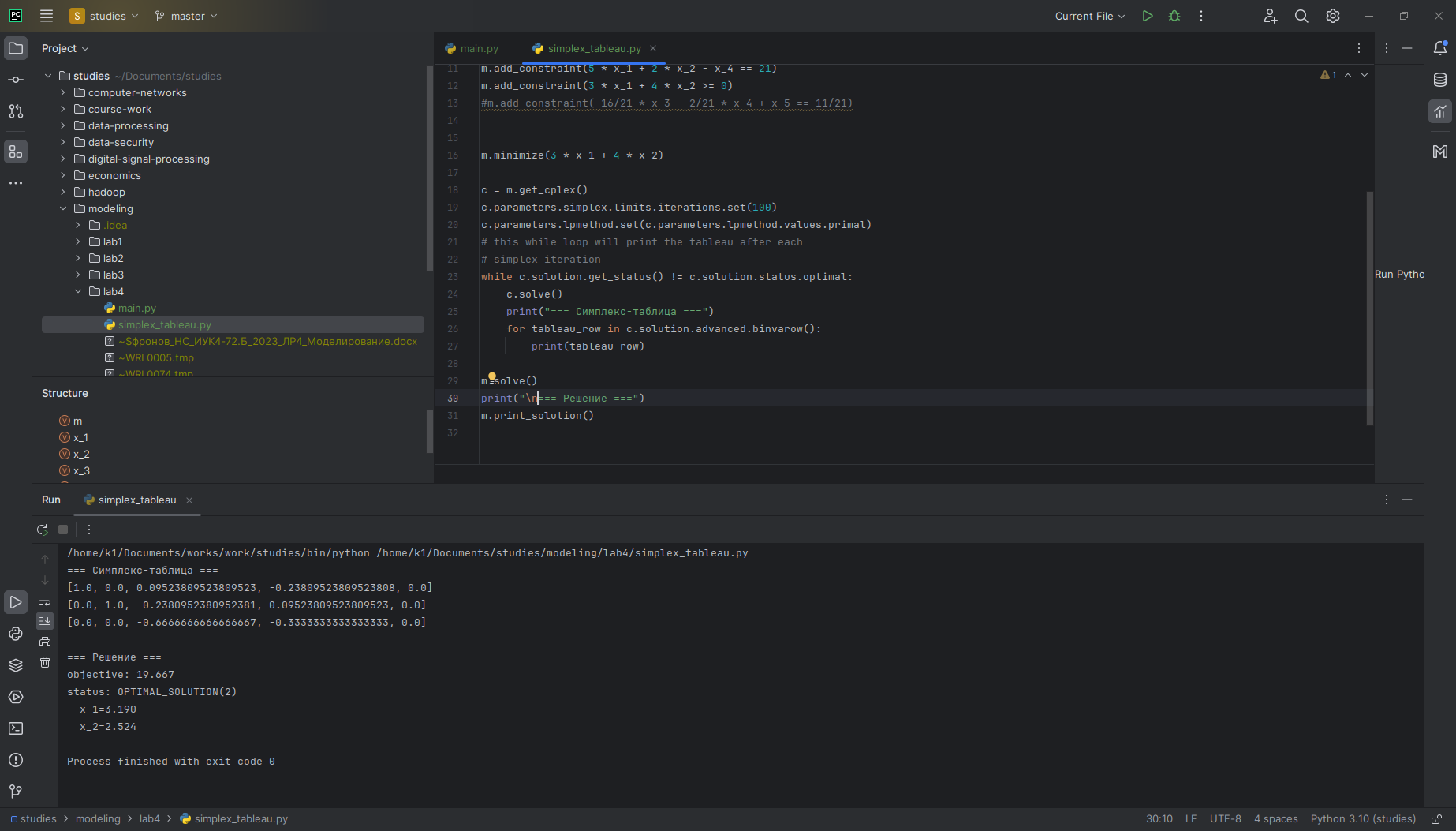
**Вариант 14**

**Ход выполнения работы**

**Первый алгоритм Гомори**

Представим задачу в канонической форме:

Решим задачу программно в произвольных числах, воспользовавшись двойственным симплекс-методом:



**Рисунок 1 –** Решение задачи в произвольных числах

Выпишем полученную симплекс-таблицу:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Базис** |  |  |  |  | **План** |
|  | 1 | 0 |  |  |  |
|  | 0 | 1 |  |  |  |
|  | 0 | 0 |  |  |  |

Получаем следующее решение в произвольных числах:

Найдём целые части оптимального решения:

Найдём дробные части оптимального решения:

Выбираем переменную с наибольшей дробной частью, т.е. .

Вводим дополнительное ограничение целочисленности:

Добавляем новую строку и получаем следующую симплекс таблицу:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Базис** |  |  |  |  |  | **План** |
|  | 1 | 0 |  |  | 0 |  |
|  | 0 | 1 |  |  | 0 |  |
|  | 0 | 0 |  |  | 1 |  |
|  | 0 | 0 |  |  | 0 |  |

Преобразуем симплекс-таблицу и получим:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Базис** |  |  |  |  |  | **План** |
|  | 1 | 0 |  |  |  |  |
|  | 0 | 1 | 0 |  |  |  |
|  | 0 | 0 |  |  |  |  |
|  | 0 | 0 |  |  |  |  |

Найдём дробные части оптимального решения:

Выбираем переменную с наибольшей дробной частью, т.е. .

Вводим дополнительное ограничение целочисленности:

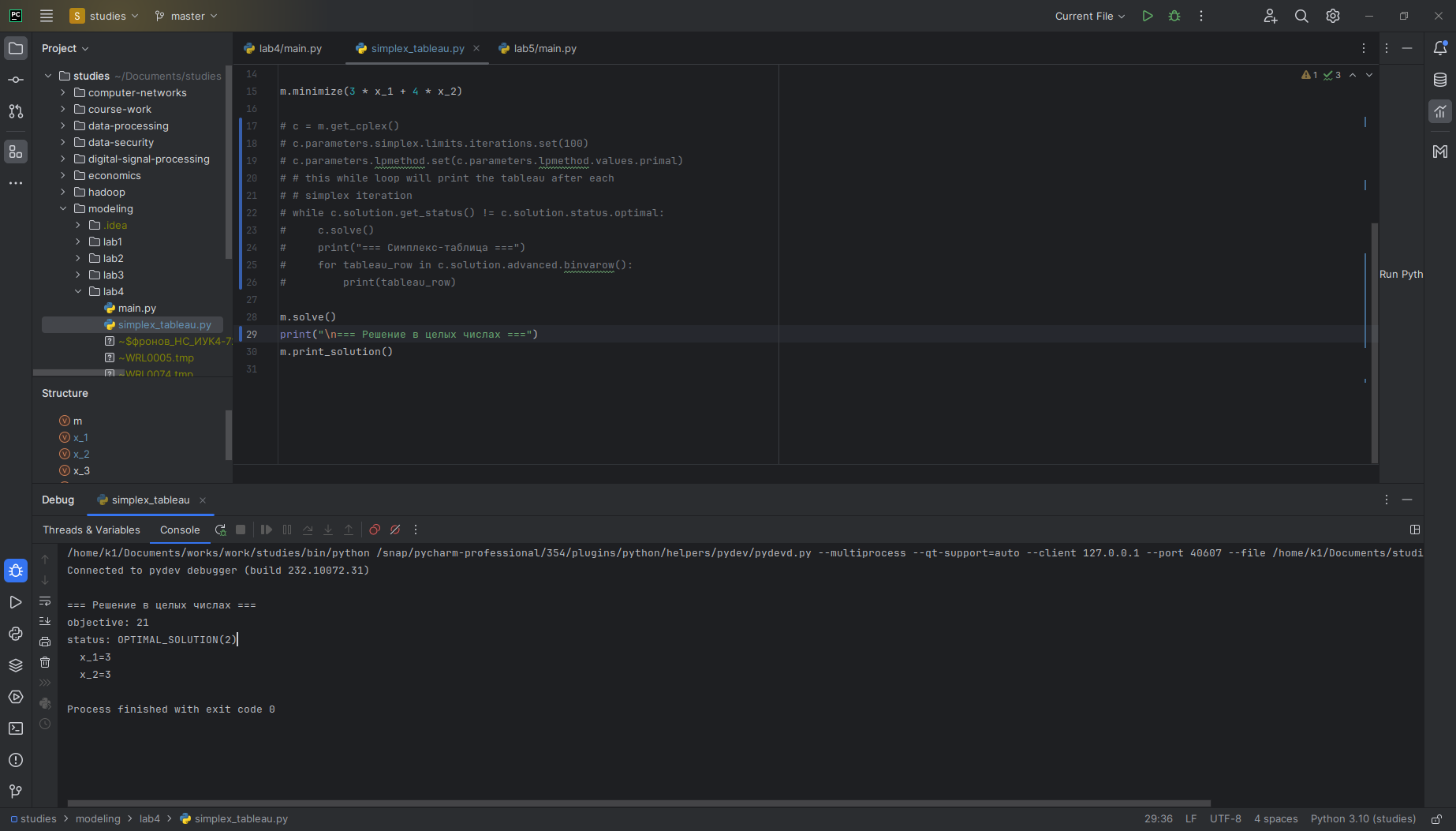
Добавляем новую строку и получаем следующую симплекс таблицу:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Базис** |  |  |  |  |  |  | **План** |
|  | 1 | 0 |  |  |  | 0 |  |
|  | 0 | 1 |  |  |  | 0 |  |
|  | 0 | 0 |  |  |  | 0 |  |
|  | 0 | 0 | 0 |  |  | 1 |  |
|  | 0 | 0 |  |  |  | 0 |  |

Преобразуем симплекс-таблицу и получим:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Базис** |  |  |  |  |  |  | **План** |
|  | 1 | 0 |  |  |  |  |  |
|  | 0 | 1 |  |  |  |  |  |
|  | 0 | 0 |  |  |  |  |  |
|  | 0 | 0 | 0 |  |  |  |  |
|  | 0 | 0 |  |  |  |  |  |

Решение получилось целочисленным. Оптимальный целочисленный план можно записать так:



**Рисунок 2 –** Решение задачи в целых числах

**Второй алгоритм Гомори**

Представим задачу в канонической форме:

Решим задачу программно в произвольных числах, воспользовавшись двойственным симплекс-методом и выпишем полученную таблицу:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Базис** |  |  |  |  | **План** |
|  | 1 | 0 |  |  |  |
|  | 0 | 1 |  |  |  |
|  | 0 | 0 |  |  |  |

В полученном оптимальном плане переменная имеет дробную часть числа. Дополнительное ограничение составляем по строке, соответствующей переменной .

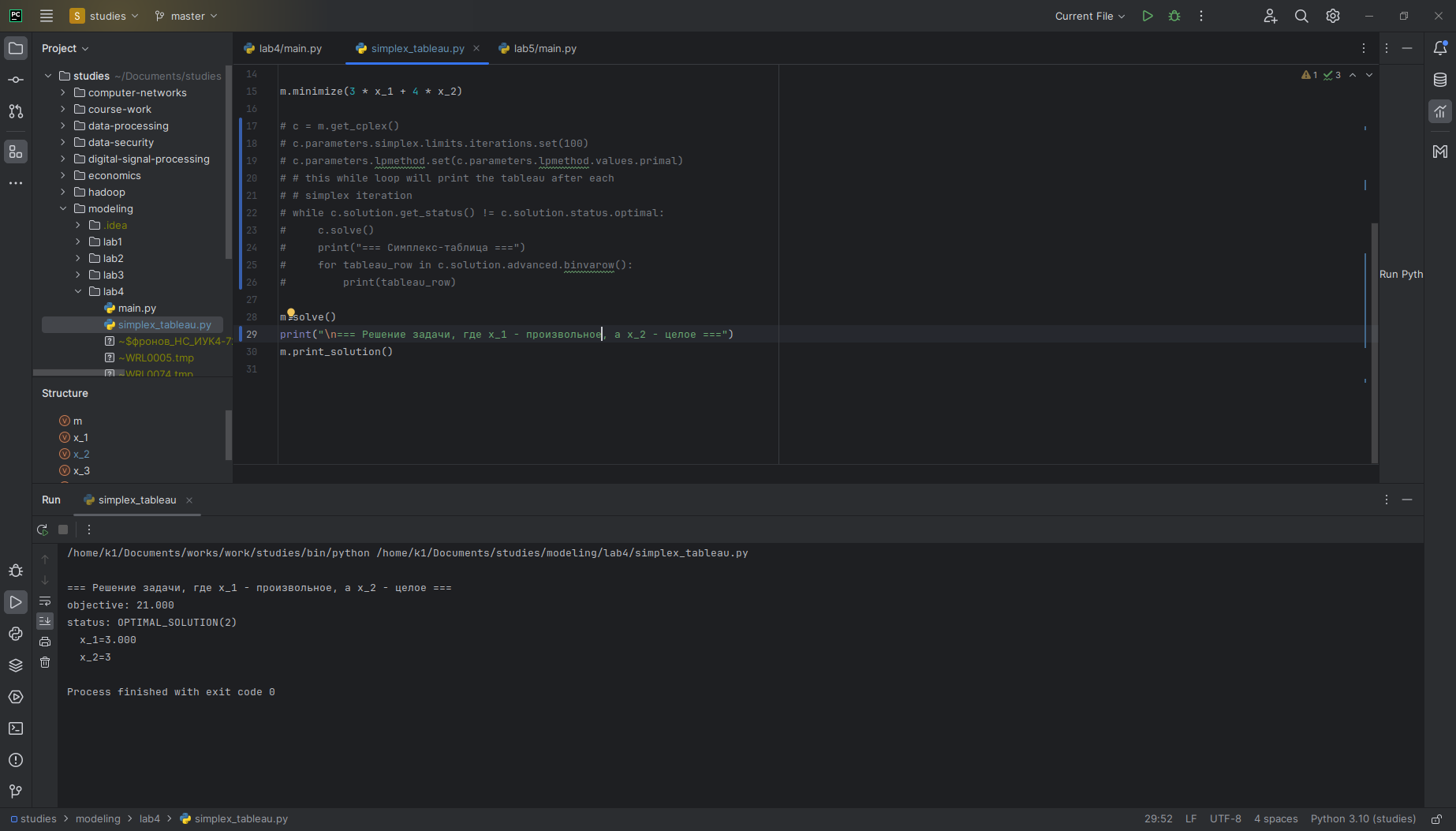
Добавляем новую строку и получаем следующую симплекс таблицу:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Базис** |  |  |  |  |  | **План** |
|  | 1 | 0 |  |  | 0 |  |
|  | 0 | 1 |  |  | 0 |  |
|  |  |  |  |  |  |  |
|  | 0 | 0 |  |  |  |  |

Преобразуем симплекс-таблицу и получим:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Базис** |  |  |  |  |  | **План** |
|  | 1 | 0 |  |  |  |  |
|  | 0 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |
|  | 0 | 0 |  |  |  |  |

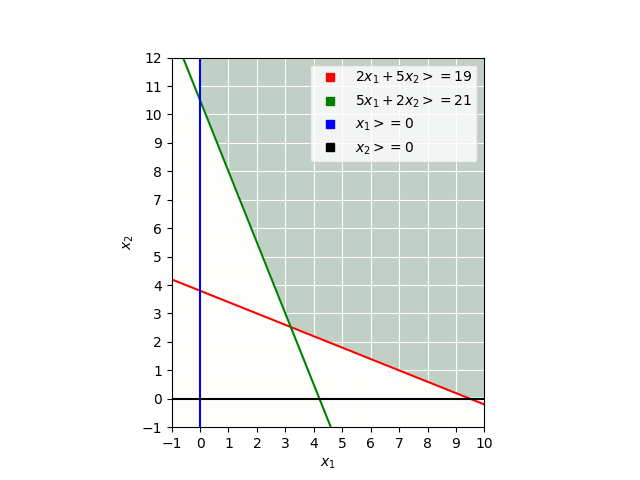
Получаем оптимальный план, который можно записать так:



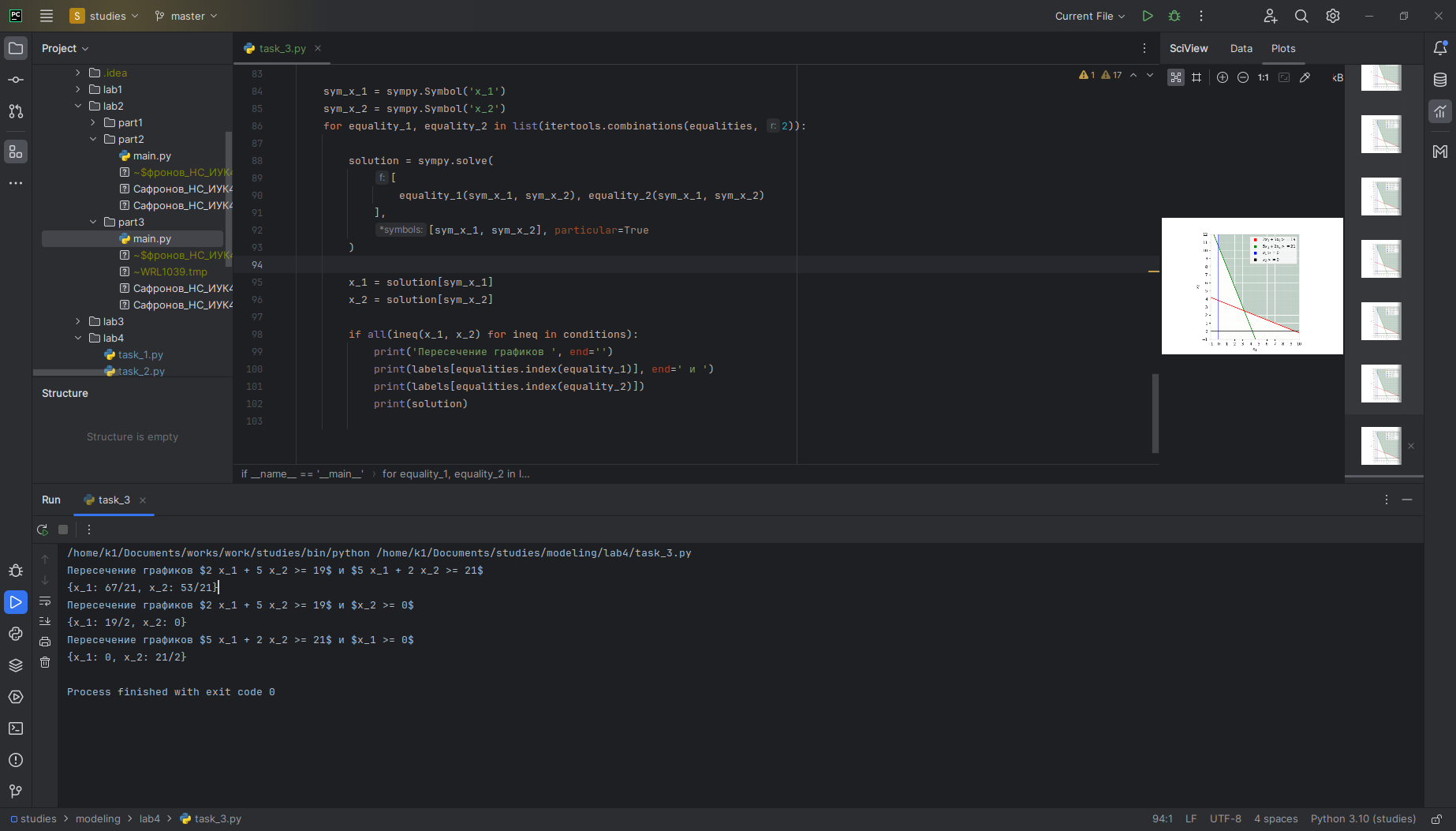
**Рисунок 3 –** Решение задачи при – произвольное, а - целое

**Метод ветвей и границ**

Построим графики, соответствующие системе:



**Рисунок 4 –** Графики, соответствующие системе



**Рисунок 5 –** Точки пересечения графиков, входящие в исследуемую область

Минимальное значение целевой функции достигается при . Будем использовать его для минимизации.

Разбиваем задачу 1 на две подзадачи 11 и 12. В первой из них к условиям задачи 11 добавляется условие , а к задаче 12 — условие .

Решим задачу 11 как задачу ЛП.

Решая задачу, получаем решение: .

Оптимальное значение переменной оказалось нецелочисленным.

Разбиваем задачу 11 на две подзадачи 111 и 112. В первой из них к условиям задачи 111 добавляется условие , а к задаче 112 — условие .

Решим задачу 111 как задачу ЛП.

Решая задачу, получаем решение: . Запоминаем значение текущего целочисленного рекорда.

Решим задачу 112 как задачу ЛП.

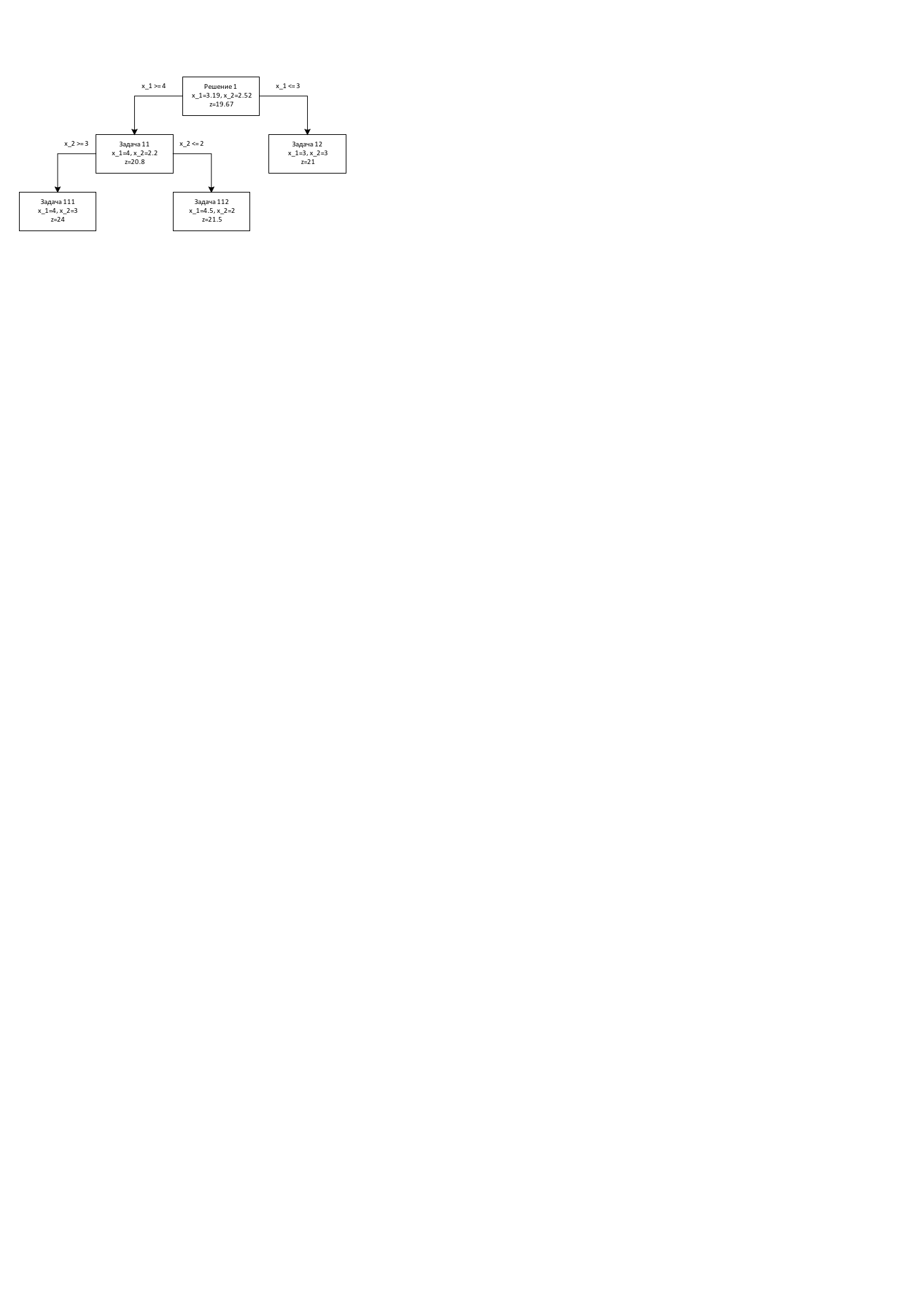
Решая задачу, получаем решение: . Оптимальное значение переменной оказалось нецелочисленным.

Решим задачу 12 как задачу ЛП.

Решая задачу, получаем решение: . Запоминаем значение текущего целочисленного рекорда .

Оптимальный план можно записать так:

.



**Рисунок 6 –** Схема метода ветвей и границ

**Вывод:** в ходе выполнения лабораторной работы были сформированы практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для стандартных вычислений при решении задач целочисленного линейного программирования на основе сравнения результатов.

**ПРИЛОЖЕНИЯ**

**Листинг программы**

**Задание 1**

from docplex.mp.model import Model  
  
m = Model()  
x\_1 = m.integer\_var(name='x\_1', lb=0)  
x\_2 = m.integer\_var(name='x\_2', lb=0)   
  
m.add\_constraint(2 \* x\_1 + 5 \* x\_2 >= 19)  
m.add\_constraint(5 \* x\_1 + 2 \* x\_2 >= 21)   
m.minimize(3 \* x\_1 + 4 \* x\_2)  
  
c = m.get\_cplex()  
c.parameters.simplex.limits.iterations.set(100)  
c.parameters.lpmethod.set(c.parameters.lpmethod.values.primal)  
  
while c.solution.get\_status() != c.solution.status.optimal:  
 c.solve()  
 print("=== Симплекс-таблица ===")  
 for tableau\_row in c.solution.advanced.binvarow():  
 print(tableau\_row)  
  
m.solve()  
print("\n=== Решение задачи ===")  
m.print\_solution()

**Задание 2**

from docplex.mp.model import Model  
  
m = Model()  
x\_1 = m.continuous\_var(name='x\_1', lb=0)  
x\_2 = m.integer\_var(name='x\_2', lb=0)   
  
m.add\_constraint(2 \* x\_1 + 5 \* x\_2 >= 19)  
m.add\_constraint(5 \* x\_1 + 2 \* x\_2 >= 21)   
  
m.minimize(3 \* x\_1 + 4 \* x\_2)

m.solve()  
print("\n=== Решение задачи, где x\_1 - произвольное, а x\_2 - целое ===")  
m.print\_solution()

**Задание 3**

import itertools  
from functools import reduce  
  
import matplotlib  
import matplotlib.pyplot as plt  
import numpy as np  
import sympy  
from matplotlib.ticker import MultipleLocator  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 conditions = [  
 lambda x\_1, x\_2: 2 \* x\_1 + 5 \* x\_2 >= 19,  
 lambda x\_1, x\_2: 5 \* x\_1 + 2 \* x\_2 >= 21,  
 lambda x\_1, x\_2: x\_1 >= 0,  
 lambda x\_1, x\_2: x\_2 >= 0  
 ]  
  
 equalities = [  
 lambda x\_1, x\_2: 2 \* x\_1 + 5 \* x\_2 - 19,  
 lambda x\_1, x\_2: 5 \* x\_1 + 2 \* x\_2 - 21,  
 lambda x\_1, x\_2: x\_1,  
 lambda x\_1, x\_2: x\_2  
 ]  
  
 labels = [  
 '$2 x\_1 + 5 x\_2 >= 19$',  
 '$5 x\_1 + 2 x\_2 >= 21$',  
 '$x\_1 >= 0$',  
 '$x\_2 >= 0$'  
 ]  
  
 colors = ['r', 'g', 'b', 'k']  
  
 x\_1\_bounds = (-1, 10)  
 x\_2\_bounds = (-1, 12)  
  
 x\_1\_range = np.linspace(x\_1\_bounds[0], x\_1\_bounds[1], 250)  
 x\_2\_range = np.linspace(x\_2\_bounds[0], x\_2\_bounds[1], 250)  
 x\_1s, x\_2s = np.meshgrid(x\_1\_range, x\_2\_range)  
  
 axis: plt.Axes  
 figure, axis = plt.subplots()  
  
 axis.set\_xlim(\*x\_1\_bounds)  
 axis.set\_ylim(\*x\_2\_bounds)  
  
 handles = []  
 for equality in equalities:  
 axis.contour(  
 x\_1s, x\_2s, equality(x\_1s, x\_2s), [0],  
 colors=colors[equalities.index(equality)]  
 )  
 handles.append(  
 matplotlib.lines.Line2D(  
 [], [], color=colors[equalities.index(equality)],  
 marker="s", ls="",  
 label=labels[equalities.index(equality)]  
 )  
 )  
  
 regions = [condition(x\_1s, x\_2s) for condition in conditions]  
 intersection = np.array(reduce(lambda \_x, \_y: \_x & \_y, regions))  
  
 extent = (x\_1s.min(), x\_1s.max(), x\_2s.min(), x\_2s.max())  
 plt.imshow(  
 intersection.astype(int),  
 extent=extent,  
 origin="lower",  
 cmap="Greens",  
 alpha=0.25  
 )  
  
 plt.xlabel("$x\_1$")  
 plt.ylabel("$x\_2$")  
  
 axis.xaxis.set\_major\_locator(MultipleLocator(1))  
 axis.yaxis.set\_major\_locator(MultipleLocator(1))  
  
 axis.grid(color='w', linestyle='-')  
  
 plt.legend(handles=handles)  
 plt.show()  
  
 sym\_x\_1 = sympy.Symbol('x\_1')  
 sym\_x\_2 = sympy.Symbol('x\_2')  
 for equality\_1, equality\_2 in list(itertools.combinations(equalities, 2)):  
  
 solution = sympy.solve(  
 [  
 equality\_1(sym\_x\_1, sym\_x\_2), equality\_2(sym\_x\_1, sym\_x\_2)  
 ],  
 [sym\_x\_1, sym\_x\_2], particular=True  
 )  
  
 x\_1 = solution[sym\_x\_1]  
 x\_2 = solution[sym\_x\_2]  
  
 if all(ineq(x\_1, x\_2) for ineq in conditions):  
 print('Пересечение графиков ', end='')  
 print(labels[equalities.index(equality\_1)], end=' и ')  
 print(labels[equalities.index(equality\_2)])  
 print(solution)