МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Информационных систем

КУРСОВАЯ РАБОТА

по дисциплине «Теория принятия решений»

Тема: Применение методов линейного и динамического программирования для решения практических задач (по вариантам)

Вариант: 110 (343)

Студент гр. 6371 _____ Шутемов А.А.

Преподаватель _____ Пономарев А.В.

Санкт-Петербург
2019

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Шутемов А.А.

Группа 6371

Тема работы: Применение методов линейного и динамического программирования для решения практических задач (по вариантам)

Исходные данные:

Задача 1

На предприятии вкладываются средства в развитие двух цехов. Функции дохода от вложенных средств для 1-го и 2-го цехов различны и представлены следующими зависимостями:

- для 1-го цеха y = 90+1.4x 3/4 ;
- для 2-го цеха y = 110+0.7x 3/4 .

где y - доход за один квартал (млн. руб); x - количество средств, вложенных за один квартал.

Функции остатка средств за один квартал равны:

- для 1-го цеха 0.95x;
- для 2-го цеха 0.77x.

Количество средств, выделяемых на развитие обоих цехов в течение года, составляет 140 единиц. Средства перераспределяются поквартально и не резервируются. Требуется оптимально распределить между двумя цехами средства на планируемый год.

Задача 2

Часть прибыли, получаемой от работы 1-го и 2-го цехов в течение одного года, планируется использовать для приобретения оборудования для нового третьего цеха. Доля средств, отчисляемая ежеквартально из прибыли от работы 1-го и 2-го цехов на приобретение оборудования для 3-го цеха, составляет 55

%. Оборудование нового цеха предполагается разместить на площади 290 кв.м. Возможно приобретение пяти видов однородного оборудования, характеристики которого представлены в таблице 1.

Таблица 1 - Характеристики оборудования

Вид оборудования	**Стоимость (млн.руб)**	**Требуемая площадь (кв.м)**	**Производительность (шт.)**
Тип 1	56	12	1000
Тип 2	25	30	500
Тип 3	53	10	1100
Тип 4	40	24	900
Тип 5	46	19	1000

Необходимо обеспечить максимальную производительность цеха и провести исследование полученного решения.

1. Имея в виду необходимость получения ЦЕЛОЧИСЛЕННОГО решения, найти оптимальный план приобретения оборудования для третьего цеха.

2. Исследовать полученное решение на чувствительность к изменению стоимостного ограничения, связанному с возможным изменением соотношения цен и средств:

- выяснить влияние изменения (увеличения, уменьшения) количества средств на переход роли активного ограничения (либо по площади, либо по стоимости), и вследствие этого — на выбор оптимального типа оборудования;
- выяснить границы изменения количества средств, в пределах которых оптимальным является выбор 2-х и более типов оборудования.

Задача 3

Выпускаемая в 3-м цехе продукция, представляющая собой полуфабрикат определенного типоразмера постоянного сечения и длиной 600 см, разрезается на заготовки длиной 450 см, 290 см, 140 см в комплектности, определяемой соотношением 1:6:9.

Требуется решить задачу оптимального раскроя в двух постановках и провести ее исследование:

1. спланировать раскрой полуфабриката, при котором число комплектов заготовок будет наибольшим;

2. спланировать раскрой полуфабриката при условии минимизации остатков и сравнить полученные результаты;

3. средствами параметрического исследования правых частей выяснить необходимое приращение количества поступивших полуфабрикатов для увеличения числа комплектов заготовок на 1 (или на 10), причем провести указанное исследование для разных значений исходного количества полуфабрикатов (проверка линейности).

Содержание пояснительной записки: «Содержание», «Введение», «1. Задача динамического программирования», «2. Задача линейного программирования», «3. Задача линейного программирования», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: _____

Дата сдачи реферата: _____

Дата защиты реферата: _____

Студент _____ Шутемов А.А.

Преподаватель _____ Пономарев А.В.

АННОТАЦИЯ

Данный курсовой проект содержит решение задач линейного и динамического программирования с использованием методов языка программирования Python. Для задач линейного программирования был использован программный пакет PuLP. Решение задачи динамического программирования использует методы библиотеки NumPy. Все графики решения визуализированы с помощью библиотеки MatPlotLib.

Работа включает 45 страниц, выполнена с привлечением 3 источников, содержит 4 рисунка, 5 таблиц и 3 приложения.

SUMMARY

This course project contains the solution of linear and dynamic programming problems using the methods of the Python programming language. For linear programming problems, the PuLP software package was used. Problem solving dynamic programming uses the methods of the library NumPy. All solution graphics are visualized using the MatPlotLib library.

The work includes 45 pages made with the involvement of 3 sources, contains 4 figures, 5 tables and 3 appendices.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: формализовать и решить программными средствами задачи линейного и динамического программирования, представленные в данном курсовом варианте.

1. ЗАДАЧА ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

1.1. Условие задачи

На предприятии вкладываются средства в развитие двух цехов. Функции дохода от вложенных средств для 1-го и 2-го цехов различны и представлены следующими зависимостями:

- для 1-го цеха: y = 90+1.4x ¾ ;
- для 2-го цеха: y = 110+0.7x ¾ .

где y – доход за один квартал (млн. руб); x – количество средств, вложенных за один квартал.

Функции остатка средств за один квартал равны:

- для 1-го цеха: 0.95x;
- для 2-го цеха: 0.77x.

Количество средств, выделяемых на развитие обоих цехов в течение года, составляет 140 единиц. Средства перераспределяются поквартально и не резервируются. Требуется оптимально распределить между двумя цехами средства на планируемый год.

1.2 Формализация задачи

Перед нами, представлена задача динамического программирования. Это можно понять из явно выраженного разделения задачи на шаги, состояния, управление и оценку.

Шаг**:** отчетный период равный кварталу (всего кварталов 4). Обозначим его через n;

Состояние**:** доход в конце квартала в зависимости от вложенных средств. Обозначим это через W_i, общий доход (назовем ее result) определяется суммой прибыли двух цехов (в нашем случае из переменных profitAx + profitBx) и с зависимостью от вложенных на следующие этапы остатка средств на конец данного квартала (остаток средств resultResidual = resudualA + resudualB). Т.е. состояние определяется как W_i(x) = profitAx + profitBx + W_{i+1}(resultResidual)

Управление**:** управлением является наибольший доход за квартал при вложенных средствах на данном и последующих шагах. Результат управления обозначим через Wmax;

Оценка**:** сравнение всей возможной прибыли в конце квартала и взятие наибольшей прибыли.

Исходя из данных, составим уравнение Беллмана:

Функция выигрыша :

$$\varphi_i(S_i) = \max(W_i(S_i))$$

$$W_i(S_i) = \frac{max}{W_i}(profitAx + profitBx + W_{i+1}(\varphi_i(\text{resultResidual}))$$

где:

- profitAx = profitA(x) – функция дохода первого цеха:
  - profitA(x) = (90+1.4*(pow(x, ¾)))
- profitBx = esidu(i-x) – функция дохода второго цеха в зависимости от оставшихся средств после вложения в первых цех;
  - esidu(x) = (110+0.7*(pow(x, ¾)))
- resultResidual = residualA + esidual – сумма остатков после работы двух цехов на конец квартала соответственно;

- residualA = fResidualA(period,resource) = pow(0.95, period)*resource – где: period – номер квартала, resource – количество средств для вычисление остатка;
- esidual = fResidualB(period,resource) = pow(0.77, period)*resource – где: period – номер квартала, resource – количество средств для вычисление остатка.

1.3. Решение задачи

Начнем с обратной прогонки.

Шаг 4.

Условный максимальный доход равен:

$W_4(S)$:

- $S_{min} = 0.77^3 * 140 = 64$;
- $S_{max} = 0.95^3 * 140 = 120$;
- $S_{linspace}$ - равномерно распределенные значения на отрезке [64,120]
- $X_{linspace}$ - равномерно распределенные значения на отрезке $[0, s \in S_{linspace}]$

$$\Rightarrow W_4(X) = 90 + 1.4\sqrt[3]{x^4} + 110 + 0.7 * \sqrt[3]{(S_{linspace} - X_{linspace})^4}$$

Результат выполнения шага представлен на рисунке 2.

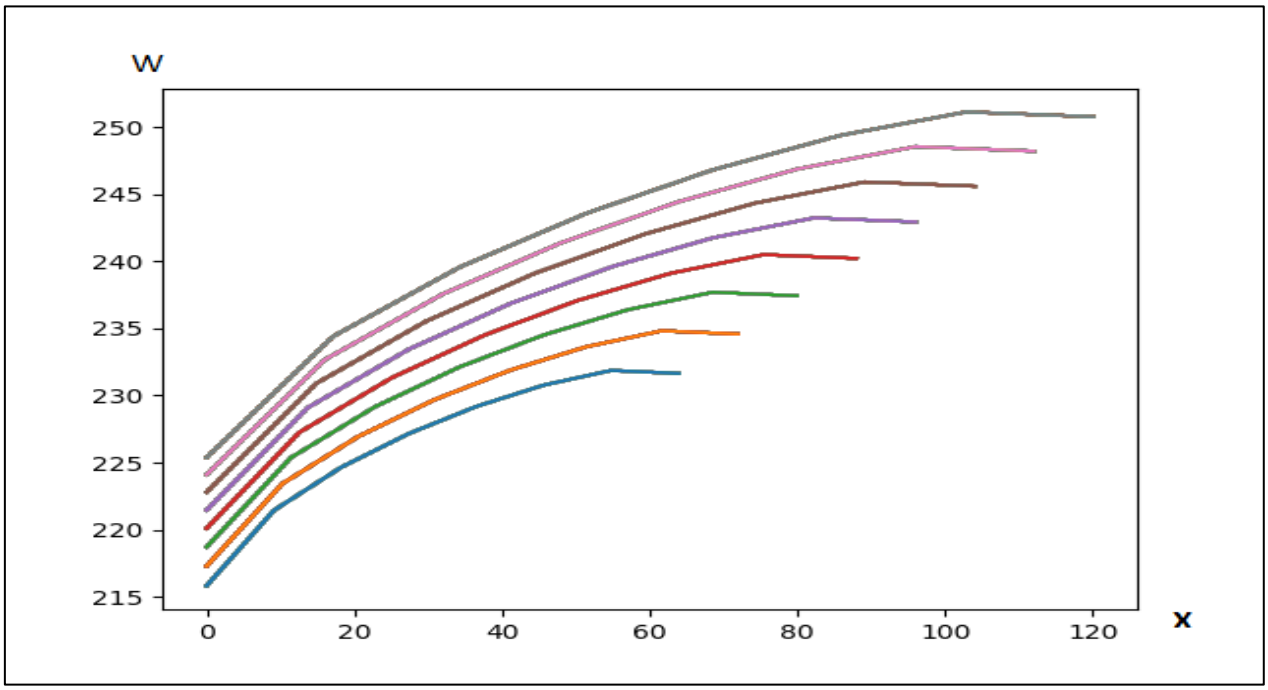

Рисунок 2 – зависимость дохода от вложенных средств на 4 шаге.

Результат выполнения программы на 4 этапе:

```
#Количество средств вкладываемых в первый цех – x.
[54.78, 61.66, 68.53, 75.4, 82.27, 89.14, 96.01, 102.88]
#Количество выделенных средств на этап – S
```

```
[63.914620000000006, 71.93146, 79.9483, 87.96513999999999, 95.98198,
103.99882, 112.01566, 120.03249999999998]
# максимальный доход
[251.13]
```

Шаг 3.

Условный максимальный доход равен:

$W_3(S)$:

- $S_{min} = 0.77^2 * 140 = 83$;
- $S_{max} = 0.95^2 * 140 = 126$;
- $S_{linspace}$ - равномерно распределенные значения на отрезке $[S_{min}, S_{max}]$
- $X_{linspace}$ – равномерно распределенные значения на отрезке $[0, s \in S_{linspace}]$

$$W_3(X) = 90 + 1.4\sqrt[3]{x^4} + 110 + 0.7 * \sqrt[3]{\left(S_{linspace} - X_{linspace}\right)^4} +$$

$+W_4(\text{residual})$ – где residual (сумма остатков от вложения средств в цеха на конец квартала)

Результат выполнения шага представлен на рисунке 3.

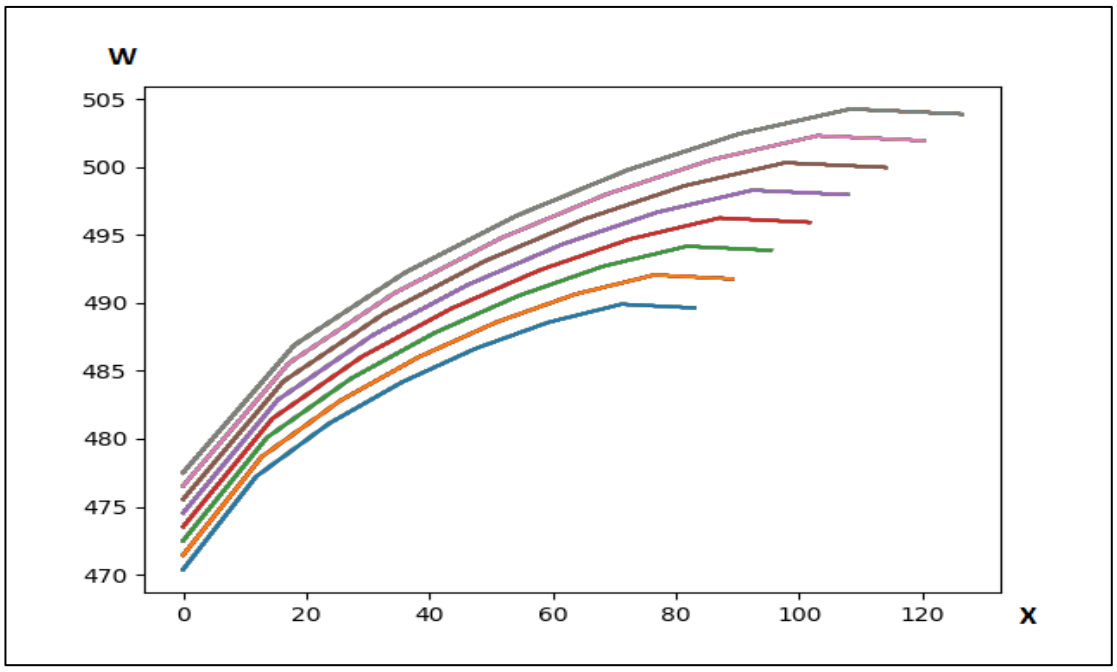

Рисунок 3 – зависимость дохода от вложенных средств на 3 шаге

Результат выполнения программы на 3 этапе:

#Количество средств вкладываемых в первый цех – x.

[71.15, 76.46, 81.76, 87.07, 92.38, 97.69, 102.99, 108.3]

#Количество выделенных средств на этап – S

[83.006, 89.198, 95.39, 101.582, 107.774, 113.966, 120.15799999999999, 126.35]

Условный максимальный доход равен:

[504.26]

Шаг 2.

Условный максимальный доход равен:

$W_2(S)$:

- S_{min} = 0.77 * 140 = 108;
- S_{max} =0.95 * 140 = 133;
- $S_{linspace}$ - равномерно распределенные значения на отрезке [S_{min}, S_{max}];
- $X_{linspace}$ – равномерно распределенные значения на отрезке [0, s ∈ $S_{linspace}$]

$$W_2(X)= 90 + 1.4\sqrt[3]{x^4} + 110 + 0.7 * \sqrt[3]{\left(S_{linspace} - X_{linspace}\right)^4} + W_3(\text{residual})$$

где residual (сумма остатков от вложения средств в цеха на конец квартала).

Результат выполнения шага представлен на рисунке 4:

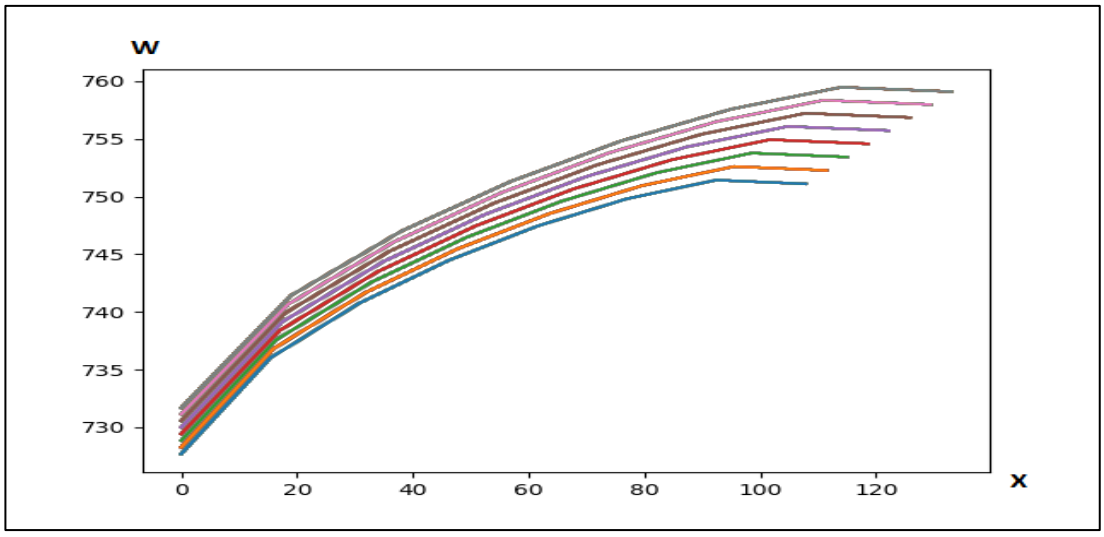

Рисунок 4 – зависимость дохода от вложенных средств на 2 шаге.

Результат выполнения программы на 2 этапе:

```
#Количество средств вкладываемых в первый цех – x.
[92.4, 95.49, 98.57, 101.66, 104.74, 107.83, 110.91, 114.0]
#Количество выделенных средств на этап – S
[107.8, 111.39999999999999, 115.0, 118.6, 122.2, 125.8, 129.4, 133.0]
# максимальный доход
[759.47]
```

ШАГ 1.

Условный максимальный доход равен:

$W_1(S)$:

- $S_{min} = 0$;
- $S_{max} = 140$;
- $S_{linspace}$ - равномерно распределенные значения на отрезке $[S_{min}, S_{max}]$;
- $X_{linspace}$ – равномерно распределенные значения на отрезке $[0, s \in S_{linspace}]$.

$$W_1(X) = 90 + 1.4\sqrt[3]{x^4} + 110 + 0.7\sqrt[3]{(S_{linspace} - X_{linspace})^4} + +W_2(\text{residual})$$

где residual (сумма остатков от вложения средств в цеха на конец квартала)

Результат выполнения шага представлен на рисунке 5.

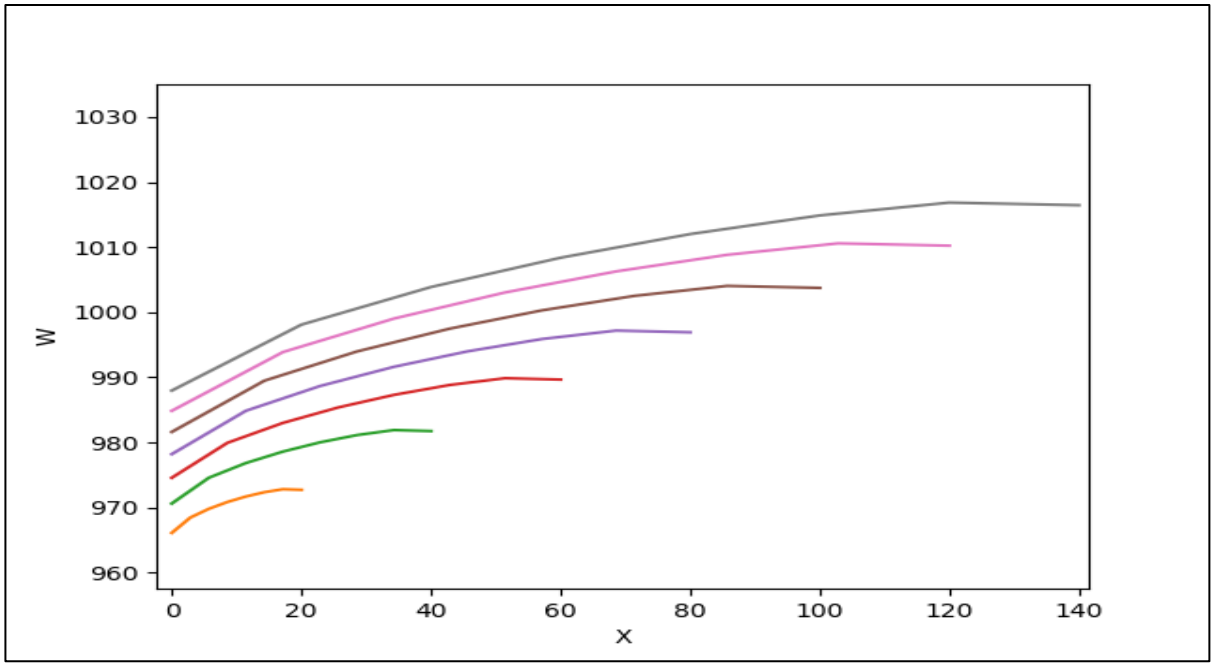

Рисунок 5 – зависимость дохода от вложенных средств на 1 шаге.

Результат выполнения программы на 1 этапе:

```
#Количество средств вкладываемых в первый цех – x.
[0.0, 17.14, 34.29, 51.43, 68.57, 85.71, 102.86, 120.0]
#Количество выделенных средств на этап – S
[0.0, 20.0, 40.0, 60.0, 80.0, 100.0, 120.0, 140.0]
# максимальный доход
[1016.85]
```

Осуществим прямую прогонку решения задачи, для этого определим на единственной кривой графика $W_1(X)$ максимум (W_{1max}= 1016.85), находим оптимальное управление на первом шаге x1=120, показывающее, сколько средств надо вкладывать в первый цех, чтобы достичь максимального дохода **,** а также количество средств вкладываемых во второй цех:

$$x1=120$$

$$x1'=k-x1 =140-120=20;$$

Находим соответствующий запас средств к концу первого шага:

$$k'=x1\cdot 0.95+(x1')\cdot 0.77=114+15.4 = 129,4$$

Входя с этим значением k' в график $x_2(k')$ найдем оптимальное управление на втором шаге, показывающее сколько средств нужно вкладывать в первый цех:

$$x2=114;$$

А также количество средств вкладываемых во второй цех:

$$x2'=k'-x2=129,4-114=15,4;$$

Находим соответствующий запас средств к концу первого шага:

$$k''=x2\cdot 0.95+(x2')\cdot 0.77=108,3 + 11,858= 120,158$$

Входя с этим значением k'' в график $x_3(k'')$ найдем оптимальное управление на втором шаге, показывающее сколько средств нужно вкладывать в первый цех:

$$x3=108,3;$$

А также количество средств вкладываемых во второй цех:

$$x3' = k'' - x3 = 120{,}158 - 108{,}4 = 11{,}858;$$

Находим соответствующий запас средств к концу 3 шага:

$$k''' = x3 \cdot 0.95 + (x3') \cdot 0.77 = 102{,}885 + 9{,}13 = 112{,}01$$

Входя с этим значением k''' в график $x_4(k''')$ найдем оптимальное управление на втором шаге, показывающее сколько средств нужно вкладывать в первый цех:

$$x4 = 102{,}88;$$

А также количество средств вкладываемых во второй цех:

$$x4' = k''' - x2 = 120{,}158 - 108{,}4 = 9{,}13566;$$

Вывод:

Исходя из вышеизложенного исследования, составим рекомендации по оптимальному распределению средств. Из имеющегося в начале квартала запаса средств k=140 усл. Ед. и остающихся средств в конце каждого квартала нужно вкладывать по кварталам в цеха I и II следующие суммы (таблица 2):

Таблица 2 – Рекомендации по оптимальному распределению средств

Кварталы	1-й	2-й	3-й	4-й
I цех	120	114	108,3	102,88
II цех	20	15,4	11,86	9,14

При таком планировании будет получена максимальная прибыль за год, равная W_1max = 1016.85 млн. руб..

Находим соответствующий запас средств к концу 4 шага:

$$k'''' = x4 \cdot 0.95 + (k''' - x4) \cdot 0.77 = 97{,}736 + 7{,}03 = 104{,}766$$

2. ЗАДАЧА ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

2.1. Условие задачи

Задача 2

Часть прибыли, получаемой от работы 1-го и 2-го цехов в течение одного года, планируется использовать для приобретения оборудования для нового третьего цеха. Доля средств, отчисляемая ежеквартально из прибыли от работы 1-го и 2-го цехов на приобретение оборудования для 3-го цеха, составляет 55 %. Оборудование нового цеха предполагается разместить на площади 290 кв.м. Возможно приобретение пяти видов однородного оборудования, характеристики которого представлены в таблице 1 исходных данных.

Необходимо обеспечить максимальную производительность цеха и провести исследование полученного решения.

1. Имея в виду необходимость получения ЦЕЛОЧИСЛЕННОГО решения, найти оптимальный план приобретения оборудования для третьего цеха.
2. Исследовать полученное решение на чувствительность к изменению стоимостного ограничения, связанному с возможным изменением соотношения цен и средств:
   2.1. выяснить влияние изменения (увеличения, уменьшения) количества средств на переход роли активного ограничения (либо по площади, либо по стоимости), и вследствие этого — на выбор оптимального типа оборудования;
   2.2. выяснить границы изменения количества средств, в пределах которых оптимальным является выбор 2-х и более типов оборудования.

2.2. Формализация задачи

2.2.1 Задание 2.1

Целевая функция:

$$z(x)=1000*x1+500*x2+1100*x3+900*x4+1000*x5 \rightarrow max$$

Ограничения:

$$S(x)= 56*x1 + 25*x2 + 53*x3 + 40*x4 + 46*x5 <= n$$

$$P(x)= 12*x1 + 30*x2 + 10*x3 + 24*x4 + 19*x5 <= 290$$

Значение ограничения стоимости получено из первой задачи следующим образом:

$$n = 1016.85*0.55 =559,2675$$

Запишем данные в формате обработки инструментом PuLP (код из приложения Б)

```
print("2.1")
problem = pulp.LpProblem('0',pulp.LpMaximize)
problem += 1000*x1+500*x2+ 1100*x3 + 900*x4 + 1000*x5 , "Целевая функция"
problem += 56*x1 + 25*x2 +53*x3+40*x4+46*x5 <= 559.2675,"1"
problem += 12*x1 + 30*x2 + 10*x3 + 24*x4 + 19*x5 <= 290, "2"
problem.solve()
```

После выполнения поиска решения получаем следующие результаты:

```
2.1
   Результат (кол. типа оборудования):
   x1 = 0.0
   x2 = 0.0
   x3 = 0.0
   x4 = 7.0
   x5 = 6.0
   Целевая функция:
   12300.0
```

Вывод: после исследования входных данных инструментом PuLP, мы можем сделать вывод о том, что доступных ресурсах оптимальным будет приобретение 7 ед. оборудования 4-го типа и 6 ед. оборудования 5-го типа.

2.2.2 Задание 2.2.1

Выясним влияние изменения (увеличения, уменьшения) количества средств на переход роли активного ограничения (в нашем случае мы выбрали изменение стоимости), и вследствие этого — на выбор оптимального типа оборудования;

Возьмем активное ограничение по стоимости:

S(x)= 56*x1 + 25*x2 + 53*x3 + 40*x4 + 46*x5 <= 559.2675

И в несколько раз увеличим количество возможных средств

S(x)= 56*x1 + 25*x2 + 53*x3 + 40*x4 + 46*x5 <= 2000

Результат выполнения программы при обновленном ограничении:

```
Результат (кол. определенного типа оборудования):
     x1 = 0.0
     x2 = 0.0
     x3 = 29.0
     x4 = 0.0
     x5 = 0.0
     Целевая функция:
     31900.0
```

Из вышеприведенных данных можно сделать вывод о том, что при ограничении площади равным 290 и условно “неограниченными” финансовыми ресурсами, оптимальным выбором будет приобретение 29 ед. оборудования 3 типа.

Так же можно определить точку перехода ограничения стоимости. Из стоимости оборудования 3 типа равным 53 млн.руб. и его количества, получаем точку 29*53 = 1537 млн.руб. .

Выполним проверку:

1) При ограничении:

S(x)= 56*x1 + 25*x2 + 53*x3 + 40*x4 + 46*x5 <= 1537

Получаем:

```
Результат (кол. определенного типа оборудования):
x1 = 0.0
```

```
x2 = 0.0
x3 = 29.0 #результат как при ограничении = 2000
x4 = 0.0
x5 = 0.0
Целевая функция:
31900.0
```

2) При ограничении меньше 1537:

$$S(x) = 56*x1 + 25*x2 + 53*x3 + 40*x4 + 46*x5 <= 1536$$

Получаем:

```
Результат (кол. определенного типа оборудования):
x1 = 0.0
x2 = 0.0
x3 = 28.0 # фиксируем изменение
x4 = 0.0
x5 = 0.0
Целевая функция:
30800.0
```

Вывод: при бесконечных средствах получаем оптимальный выбор равным 29 ед. оборудования 3 типа. 1537 млн.руб. точка влияния на оптимальный выбор, меньше - не эффективно , больше - не выгодно.

2.2.3 Задание 2.2.2

Выясним границы изменения количества средств, в пределах которых оптимальным является выбор 2-х и более типов оборудования.

Для этого будем изменять количество стоимостных средств в пределах [0,1537], и фиксировать количество средств, когда оптимальным будет являться выбор более одного типа оборудования.

Часть программного кода, обеспечивающий поиск границ (код из приложения Б):

```
problem.solve()
for i in range(1537):
problem = pulp.LpProblem('0', pulp.LpMaximize)
    problem += 1000 * x1 + 500 * x2 + 1100 * x3 + 900 * x4 + 1000 * x5,
"Целевая функция"
    problem += 56 * x1 + 25 * x2 + 53 * x3 + 40 * x4 + 46 * x5 <= i, "1"
problem += 12 * x1 + 30 * x2 + 10 * x3 + 24 * x4 + 19 * x5 <= 290, "2"
    problem.solve()
    if(problem.variables()[0].varValue!=0 and
problem.variables()[1].varValue==0 and problem.variables()[2].varValue==0 and
problem.variables()[3].varValue==0 and problem.variables()[4].varValue==0):
        continue
    elif(problem.variables()[0].varValue==0 and
problem.variables()[1].varValue!=0 and problem.variables()[2].varValue==0 and
problem.variables()[3].varValue==0 and problem.variables()[4].varValue==0):
        continue
    elif(problem.variables()[0].varValue==0 and
problem.variables()[1].varValue==0 and problem.variables()[2].varValue!=0 and
problem.variables()[3].varValue==0 and problem.variables()[4].varValue==0):
        continue
    elif(problem.variables()[0].varValue==0 and
problem.variables()[1].varValue==0 and problem.variables()[2].varValue==0 and
problem.variables()[3].varValue!=0 and problem.variables()[4].varValue==0):
        continue
    elif(problem.variables()[0].varValue==0 and
problem.variables()[1].varValue==0 and problem.variables()[2].varValue==0 and
problem.variables()[3].varValue==0 and problem.variables()[4].varValue!=0):
```

```
        continue
    elif(problem.variables()[0].varValue==0 and
problem.variables()[1].varValue==0 and problem.variables()[2].varValue==0 and
problem.variables()[3].varValue==0 and problem.variables()[4].varValue==0):
        continue
    else:
        test.append(i)
```

В данном примере рассматриваем случаи, когда мы не ограничиваемся выбором единственного типа оборудования.

Результат выполнения программы:

- *[65, 66, 67, 68, 69, 70, 71, 72, 73, 74]*
- *[78, 79]*
- *[86, 87, 88, 89, 90, 91]*
- *[96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119]*
- *[126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137]*
- *[140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159]*
- *[166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183]*
- *[185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199]*
- *[206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239]*
- *[246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279]*
- *[286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319]*
- *[326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359]*

- *[366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399]*

- *[406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439]*

- *[446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479]*

- *[486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597]*

- *[601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643]*

- *[651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689]*

- *[697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027,*

1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271]

- *[1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324]*

- *[1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377]*

- *[1403, 1404, 1405, 1406, 1407, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430]*

- *[1477, 1478, 1479, 1480, 1481, 1482, 1483]*

Вывод: было произведено исследование, благодаря которому мы получили результат равный 21 диапазону количества средств, в каждом из которых оптимальным является приобретение более одного типа оборудования.

3. ЗАДАЧА ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

3.1. Условие задачи

Задача 3.

Выпускаемая в 3-м цехе продукция, представляющая собой полуфабрикат определенного типоразмера постоянного сечения и длиной 600 см, разрезается на заготовки длиной 450 см, 290 см, 140 см в комплектности, определяемой соотношением 1:6:9.

Требуется решить задачу оптимального раскроя в двух постановках и провести ее исследование:

1. спланировать раскрой полуфабриката, при котором число комплектов заготовок будет наибольшим;

2. спланировать раскрой полуфабриката при условии минимизации остатков и сравнить полученные результаты;

3. средствами параметрического исследования правых частей выяснить необходимое приращение количества поступивших полуфабрикатов для увеличения числа комплектов заготовок на 1 (или на 10), причем провести указанное исследование для разных значений исходного количества полуфабрикатов (проверка линейности).

3.2. Формализация задачи

3.2.1 Задание 3.1

Спланируем раскрой полуфабриката, при котором число комплектов заготовок будет наибольшим. Для начала сформируем общие сведения о задаче в таблице 3, а затем в таблице 4 представим варианты раскроя:

Таблица 3 - Общие сведения о задаче

Длина 1 ед. полуфабриката	600 см
Кол. Полуфабриката для задания 3.1	100 ед.
Длина заготовки 1	450 см
Длина заготовки 2	290 см
Длина заготовки 3	140 см

Таблица 4 - Варианты раскроя

	X1	X2	X3	X4	Result
Problem 1	1	0	0	-1	0
Problem 2	0	2	1	-6	0
Problem 3	1	0	2	-9	0
Problem 4	1	1	1		<=100

Сформируем целевую функцию:

$$z(x)=1*x4 \rightarrow max$$

Ограничения:

$$problem1 = 1*x1 + 0*x2 + 0*x3 - 1*x4 == 0$$

$$problem2 = 0*x1 + 2*x2 + 1*x3 - 6*x4 == 0$$

$$problem3 = 1*x1 + 0*x2 + 2*x3 - 9*x4 == 0$$

$$problem4 = 1*x1 + 1*x2 + 1*x3 <= 100$$

Форматируем ограничения для дальнейшей обработкой инструментом PuLP (код из приложения В):

```
problem = pulp.LpProblem('0',pulp.LpMaximize)
problem += x4 ,"Целевая функция"
problem += 1*x1 + 0*x2 + 0*x3 - 1*x4 == 0,"1"
problem += 0*x1 + 2*x2 + 1*x3 - 6*x4 == 0, "2"
problem += 1*x1 + 0*x2 + 2*x3 - 9*x4 == 0, "3"
problem += 1*x1 + 1*x2 + 1*x3  <= 100, "4"
```

```
problem.solve()
```

Результат работы программы представлен ниже:

```
3.1
Результат:
x1 = 16.0
x2 = 16.0
x3 = 64.0
x4 = 16.0
Целевая функция:
16.0
```

Вывод:

При ограничении в 100 ед. полуфабриката, целевая функция показала результат равный 16 ед. изготовленных комплектов.

3.2.2 Задание 3.2

Спланируем раскрой полуфабриката при условии минимизации остатков и сравним полученные результаты с результатами задания 3.1;

Варианты раскроя остаются прежними (таблица 4),

Целевая функция примет вид:

$$z(x)= 10*x1 + 20*x2 + 30*x3 \rightarrow \min;$$

где {x1,x2,x3} – количество остатков при варианте раскроя, соответственно.

Ограничения:

problem1 = 1*x1 + 0*x2 + 0*x3 - 1*x4 == 0

problem2 = 0*x1 + 2*x2 + 1*x3 - 6*x4 == 0

problem3 = 1*x1 + 0*x2 + 2*x3 - 9*x4 == 0

problem4 = 1*x1 + 1*x2 + 1*x3 == 100 (кол. полуфабриката)

Форматируем ограничения для дальнейшей обработкой инструментом PuLP (код из приложения В):

```
problem = pulp.LpProblem('0',pulp.LpMinimize)
problem += 10*x1 + 20*x2 + 30*x3 ,"ostatok"
problem += 1*x1 + 0*x2 + 0*x3 - 1*x4 == 0,"1"
problem += 0*x1 + 2*x2 + 1*x3 - 6*x4 == 0, "2"
problem += 1*x1 + 0*x2 + 2*x3 - 9*x4 == 0, "3"
problem += 1*x1 + 1*x2 + 1*x3  == 100,"4"
problem.solve()
```

Результат выполнения программы:

```
3.2
Результат:
x1 = 16.666667
x2 = 16.666667
x3 = 66.666667
x4 = 16.666667
Целевая функция:
2500.0000200000004
```

Вывод:

Из результатов видно, что целочисленное количество раскроек всех типов равно целочисленному количеству раскроек всех типов из задания 3.1.. Следовательно можно сделать вывод о том, что количество комплектов так же равно (16 ед.).

3.2.3 Задание 3.3.

Целевая функция примет вид:

z(x)= 1*x1 + 1*x2 + 1*x3 -> max; (кол. полуфабриката)

Ограничения:

problem1 = 1*x1 + 0*x2 + 0*x3 - 1*x4 == 0

problem2 = 0*x1 + 2*x2 + 1*x3 - 6*x4 == 0

problem3 = 1*x1 + 0*x2 + 2*x3 - 9*x4 == 0

problem4 = 1*x4 == n (кол. комплектов)

С помощью программного инструмента PuLP, выясним необходимое приращение количества полуфабрикатов (z) для увеличения числа комплектов заготовок на 1, проверим линейность.

Для этого найдем необходимое количество полуфабриката для производства n ($n \in [0,20]$) заготовок.

Форматируем ограничения для дальнейшей обработкой инструментом (код из приложения В):

```
for i in range(21):
    problem = pulp.LpProblem('0', pulp.LpMaximize)
    global bugger
    problem += 1*x1 + 1*x2 + 1*x3  , "целевая функция"
    problem += 1*x1 + 0*x2 + 0*x3 - 1*x4 == 0,"1"
    problem += 0*x1 + 2*x2 + 1*x3 - 6*x4 == 0, "2"
    problem += 1*x1 + 0*x2 + 2*x3 - 9*x4 == 0, "3"
    problem += 1*x4 ==i, "4"
    problem.solve()
```

Результат выполнения программы:

```
[0.0, 6.0, 12.0, 18.0, 24.0, 30.0, 36.0, 42.0, 48.0, 54.0, 60.0, 66.0,
72.0, 78.0, 84.0, 90.0, 96.0, 102.0, 108.0, 114.0, 120.0]
```

Представим результат в табличной форме (таблица 5):

Таблица 5 - результаты работы программы 3.3.

Количество комплектов.	Требуемое количество полуфабриката
1	6
2	12
3	18
4	24

Продолжение таблицы 5 - результаты работы программы 3.3.

Количество комплектов.	Требуемое количество полуфабриката
5	30
6	36
7	42
8	48
9	54
10	60
11	66
12	72
13	78
14	84
15	90
16	96
17	102
18	108
19	114
20	120

Вывод:

Результат выполнения программы говорит о том, что при увеличении количества комплектов заготовок на 1 (начиная с нуля), нам потребуется добавлять по 6 ед. полуфабриката на изготовление нового комплекта заготовок. Из результатов видно, что при этом сохраняется линейность требуемых полуфабрикатов.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе были проанализированы и формализированы задачи динамического и линейного программирования, исследование которых было произведено с использованием языка программирования Python в совокупности с библиотеками MatPlotLib и PuLP.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Пономарев А.В. Динамическое программирование с помощью GNU Octave за 7 простых шагов // Теория принятия решений – тематический сайт. URL: https://avponomarev.bitbucket.io/DP_Octave.pdf.

2. Пономарев А.В. Решение задач линейного программирования с использованием GNU Octave, GLPK и Python // Теория принятия решений – тематический сайт. URL: https://avponomarev.bitbucket.io/LP_tutorial.pdf.

3. ГОСТ 7.32–2001. Межгосударственный стандарт. Отчет о научно-исследовательской работе. Структура и правила оформления. М.: Изд-во стандартов, 2001.

ПРИЛОЖЕНИЕ А.
КОД РЕШЕНИЯ ЗАДАЧИ 1 ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

```
import matplotlib as plt

import numpy as np

import matplotlib.pyplot as plt
import matplotlib.pyplot as plt1


# des - descritisation
    # Sv - ресурс
    # Xv - возможные варианиты распределения ресурса
    # fResidualA() - функция остатка
    # profitA() - функция дохода
    # Smin, Smax - возможный диапазон распределения ресурса на этап
  е в зависимости от функции остатка

    mainResouce = 140

    # словарь для мемоизации
    cache = {}

    des = 8

    discretS = des
    discretX = des

    mass1Q = []
    mass2Q = []
    mass3Q = []
    mass4Q = []

    mass1X = []
    mass2X = []
    mass3X = []
    mass4X = []

    mass1I = []
    mass2I = []
    mass3I = []
    mass4I = []

    Wmaxs1 = []
    Wmaxs2 = []
    Wmaxs3 = []
```

```
    Wmaxs4 = []

    def profitA(x):
        print("Profit a x:")
        print(x)
        return (90 + 1.4*(pow(x,3/4)))

    def profitB(x):
        print("Profit b x:")
        print(x)
        return (110 + 0.7*(pow(x,3/4)))

    def fResidualA(period,localRes):
        return pow(0.95,period)*localRes

    def fResidualB(period,localRes):
        return pow(0.77,period)*localRes


    def termFunc(Sv):
        print("after test start term")
        print(Sv)
        Smin = fResidualB(3,mainResouce)
        Smax = fResidualA(3,mainResouce)
        Sdistance = np.linspace(Smin, Smax, discretS)
        print("Smin" + str(Smin))
        print("Smax" + str(Smax))

        Xmaxs = []
        IintoX = []
        print("SDis" + str(Sdistance))

        AllW = []
        Wmax = []

        for i in Sdistance:
            Xv = np.linspace(0, i, discretX)
            for x in Xv:

                # главный кеш мемоизации
                global cache
                if ((4, Sv) in cache):
                    print("терминальный кеш от средств "+str(Sv))
                    return cache[4, Sv]

                profitAx = profitA(x)
                profitBx = profitB(i - x)

                if ((profitAx >= 0) and (profitBx >= 0)):
                    AllW.append(profitAx+profitBx)
```

```
94.                 else:
95.                     AllW.append(0)
96.             plt.figure(4)
97.             plt.plot(Xv, AllW)
98.             Wmax.append(round(max(AllW), 2))
99.             Xmax = Xv[AllW.index(max(AllW))]
100.            Xmaxs.append(round(Xmax, 2))
101.            IintoX.append(i)
102.            global mass4X
103.            global mass4I
104.            global mass4Q
105.            mass4Q = max(Wmax[:])
106.            mass4I = IintoX[:]
107.            mass4X = Xmaxs[:]
108.            AllW = []
109.        global Wmaxs4
110.        Wmaxs4.append(max(Wmax))
111.        cache[4, Sv] = max(Wmaxs4[:])
112.        print("Term before func result")
113.        print(Wmax)
114.        print("return term Wmax" + str(max(Wmax)))
115.        return cache[4,Sv]
116.
117.    def mainF(n,Sv):
118.        if(n==4):
119.            if (Sv <= 0.0):
120.                return 0
121.            print(Sv)
122.
123.            Sv = round(Sv, 4)
124.            print("start main function quarter " + str(n))
125.
126.            result = termFunc(Sv)
127.            print("start mainF 4")
128.            print(result)
129.            return result
130.        else:
131.                if (Sv <= 0.0):
132.                    return 0
133.                global cache
134.                print("start main function quarter " + str(n))
135.                AllW = []
136.                Wmax = []
137.                Xmaxs = []
138.                IintoX = []
139.                if(n==1):
140.                    Sv = round(Sv, 4)
141.    # на первом этапе мы можем вложить в первый цех от 1 до 140
142.                    Smin = 0
143.                    Smax = 140
```

```
144.                     Sdistance = np.linspace(Smin, Smax, discretS)
145.                 else:
146.                     Sv = round(Sv, 4)
147.                     # расчет возможных вкладов в следующий этап
148.                     Smin = fResidualB(n-1, mainResouce)
149.                     Smax = fResidualA(n-1, mainResouce)
150.                     Sdistance = np.linspace(Smin, Smax, discretS)
151.                 print("Smax " + str(Smax))
152.                 print("Smin " + str(Smin))
153.                 print("Sdis " + str(Sdistance))
154.
155.
156.                 for i in Sdistance:
157.                     # мы можем вложить в первый цех x средств от 0
  до i, во второй i-x средств
158.                     Xv = np.linspace(0, i, discretX)
159.                     for x in Xv:
160.
161.                         # МЕМОИЗАЦИЯ
162.                         print("Test before cache")
163.                         if (((n,Sv) in cache) and n!=1):
164.                             return cache[n, Sv]
165.                         else:
166.
167.                             profitAx = profitA(x)
168.                             profitBx = profitB(i-x)
169.
170.                             if((profitAx>=0) and (profitBx>=0)):
171.                                 if(n==1):
172.                                     resudualA = fResidualA(n , x)
173.                                     resudualB = fResidualB(n , i -
  x)
174.                                     resultResidual = resudualA + re
  sudualB
175.                                 else:
176.                                     resudualA = fResidualA(n-
  1, x)
177.                                     resudualB = fResidualB(n-
  1, i - x)
178.                                     resultResidual = resudualA + re
  sudualB
179.                                 # тут мы считаем доход на этапе + о
  тдаем на след. этап остаточные средства после текущего этапа.

180.                                 AllW.append(round((profitAx + profi
  tBx + mainF(n+1,resultResidual)),3))
181.                             else:
182.                                 print("NEGATIVE PROFIT ON STEP " +
  str(n))
```

```
                                # если один из цехов отдает отрицат
  ельный доход, то мы обнуляем весь доход.
                                AllW.append(0)

                    #внутри for i (Для каждого возможного вложения
  в следующий этап (Sdistance) пробегаемся по возможным вкладам x  в
  первый цех от 0 до (i))

                    #Рисуем графики для каждого этапа
                    if(n==1):
                        plt.figure(n)
                        plt.plot(Xv,AllW)
                    if (n == 2):
                        plt.figure(n)
                        plt1.plot(Xv, AllW)
                    if (n == 3):
                        plt.figure(n)
                        plt.plot(Xv, AllW)

                    Wmax.append(round(max(AllW), 2))

                    # из всех Доходов выбираем максимальный, фиксир
  уем при нем икс
                    Xmax = Xv[AllW.index(max(AllW))]

                    IintoX.append(i)
                    # добавляем xmax в массив максимумов
                    Xmaxs.append(round(Xmax, 2))

                    # Буфер значений для графики
                    if (n == 1):
                        global mass1X
                        global mass1I
                        global mass1Q
                        mass1Q = max(Wmax[:])
                        mass1I.append(i)
                        mass1X = Xmaxs[:]
                    if (n == 2):
                        global mass2X
                        global mass2I
                        global mass2Q
                        mass2Q = max(Wmax[:])
                        mass2I = IintoX[:]
                        mass2X = Xmaxs[:]
                    if (n == 3):
                        global mass3X
                        global mass3I
                        global mass3Q
                        mass3Q = max(Wmax[:])
                        mass3I = IintoX[:]
```

```
                        mass3X = Xmaxs[:]
    # обнуляем все доходы на возможном этапе i in Sdist
                    AllW = []

                if(n==1):
                    global  Wmaxs1
                    Wmaxs1.append(max(Wmax))
                    cache[n, Sv] = max(Wmaxs1[:])
                if (n == 2):
                    global Wmaxs2
                    Wmaxs2.append(max(Wmax))
                    cache[n, Sv] = max(Wmaxs2[:])
                if (n == 3):
                    global Wmaxs3
                    Wmaxs3.append(max(Wmax))
                    cache[n, Sv] = max(Wmaxs3[:])

                print("RETURN MAIN F QUARTER " + str(n))
                if(n == 1):
                    print("")
                elif(n==2):
                    return max(Wmaxs2)
                elif(n == 3):
                    return max(Wmaxs3)


mainF(1,mainResouce)

print("X")
print(mass1X)
print("I")
print(mass1I)
print("Q")
print(mass1Q)
print("_____")
print(mass2X)
print(mass2I)
print(mass2Q)
print("_____")
print(mass3X)
print(mass3I)
print(mass3Q)
print("_____")
print(mass4X)
print(mass4I)
print(mass4Q)

print("_____")
print(max(Wmaxs1))
print(max(Wmaxs2))
```

```
279.  print(max(Wmaxs3))
280.  print(max(Wmaxs4))
281.
282.  plt.xlabel('X')
283.  plt.ylabel('W')
284.
285.  plt.show()
286.
287.  pltl.xlabel('X')
288.  pltl.ylabel('W')
289.  plt1.show()
```

ПРИЛОЖЕНИЕ Б.
КОД РЕШЕНИЯ ЗАДАЧИ 2 ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

```
from pulp import *
import time
import numpy as np
start = time.time()

x1 = pulp.LpVariable("x1", lowBound=0,cat = 'Integer')
x2 = pulp.LpVariable("x2", lowBound=0,cat = 'Integer')
x3 = pulp.LpVariable("x3", lowBound=0,cat = 'Integer')
x4 = pulp.LpVariable("x4", lowBound=0,cat = 'Integer')
x5 = pulp.LpVariable("x5", lowBound=0, cat = 'Integer')

# # 2.1
print("2.1")
problem = pulp.LpProblem('0',pulp.LpMaximize)

problem += 1000*x1  + 500*x2  +1100*x3 + 900*x4 + 1000*x5 , "Функция цели"
problem += 56*x1 + 25*x2 +53*x3+40*x4+46*x5 <= 559.2675,"1"

problem += 12*x1 + 30*x2 + 10*x3 + 24*x4 + 19*x5 <= 290, "2"


problem.solve()
print ("Результат (кол. определенного типа оборудования):")
for variable in problem.variables():
    print (variable.name, "=", variable.varValue)
print ("Целевая функция:")
print (abs(value(problem.objective)))
stop = time.time()
print ("Время :")
print(stop - start)

# исследование 2.2.1
# при бесконечных средствах получаем оптимальный выбор = 29 ед 3 типа
# максимальный 29*53 = 1537 точка влияния на оптимальный выбор, меньше - не эффективно , больше - не выгодно
problem = pulp.LpProblem('0',pulp.LpMaximize)

problem += 1000*x1  + 500*x2  +1100*x3 + 900*x4 + 1000*x5 , "Функция цели"
problem += 56*x1 + 25*x2 +53*x3+40*x4+46*x5 <= 1537,"1"
problem += 12*x1 + 30*x2 + 10*x3 + 24*x4 + 19*x5 <= 290, "2"

```

```
41.
42. problem.solve()
43. print ("")
44. print ("Результат (кол. определенного типа оборудования):")
45. for variable in problem.variables():
46.     print (variable.name, "=", variable.varValue)
47. print ("Целевая функция:")
48. print (abs(value(problem.objective)))
49. stop = time.time()
50. print ("Время :")
51. print(stop - start)
52.
53. # исследование 2.2.2
54. print("2.2.2")
55. print ("")
56. print ("результат")
57. test = []
58. for i in range(1537):
59.     problem = pulp.LpProblem('0', pulp.LpMaximize)
60.     problem += 1000 * x1 + 500 * x2 + 1100 * x3 + 900 * x4 + 1000 *
     x5, "Функция цели"
61.     problem += 56 * x1 + 25 * x2 + 53 * x3 + 40 * x4 + 46 * x5 <= i
    , "1"  # 86 - Значение
62.     problem += 12 * x1 + 30 * x2 + 10 * x3 + 24 * x4 + 19 * x5 <= 2
    90, "2"
63.     problem.solve()
64.     buffer = problem.variables()[:]
65.
66.     if(problem.variables()[0].varValue!=0 and problem.variables()[1
    ].varValue==0 and problem.variables()[2].varValue==0 and problem.va
    riables()[3].varValue==0 and problem.variables()[4].varValue==0):
67.         continue
68.     elif(problem.variables()[0].varValue==0 and problem.variables()
    [1].varValue!=0 and problem.variables()[2].varValue==0 and problem.
    variables()[3].varValue==0 and problem.variables()[4].varValue==0):

69.         continue
70.     elif(problem.variables()[0].varValue==0 and problem.variables()
    [1].varValue==0 and problem.variables()[2].varValue!=0 and problem.
    variables()[3].varValue==0 and problem.variables()[4].varValue==0):

71.         continue
72.     elif(problem.variables()[0].varValue==0 and problem.variables()
    [1].varValue==0 and problem.variables()[2].varValue==0 and problem.
    variables()[3].varValue!=0 and problem.variables()[4].varValue==0):

73.         continue
74.     elif(problem.variables()[0].varValue==0 and problem.variables()
    [1].varValue==0 and problem.variables()[2].varValue==0 and problem.
```

```
    variables()[3].varValue==0 and problem.variables()[4].varValue!=0):
75.         continue
76.     elif(problem.variables()[0].varValue==0 and problem.variables()
    [1].varValue==0 and problem.variables()[2].varValue==0 and problem.
    variables()[3].varValue==0 and problem.variables()[4].varValue==0):
77.         continue
78.     else:
79.         test.append(i)
80.
81. print("COST")
82. print(test)
```

ПРИЛОЖЕНИЕ В.
КОД РЕШЕНИЯ ЗАДАЧИ 3 ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

```
from pulp import *
import time
import numpy as np
start = time.time()

x1 = pulp.LpVariable("x1", lowBound=0,cat = 'Integer')
x2 = pulp.LpVariable("x2", lowBound=0,cat = 'Integer')
x3 = pulp.LpVariable("x3", lowBound=0,cat = 'Integer')
x4 = pulp.LpVariable("x4", lowBound=0,cat = 'Integer')
    x5 = pulp.LpVariable("x5", lowBound=0, cat = 'Integer')
    x6 = pulp.LpVariable("x6", lowBound=0, cat = 'Integer')

    # всего 600
    # 1 - 450  1                   450
    # 2 - 290  2 + 4 = 6          1740
    # 3 - 140  8+1 = 9 ; 140*9 = 1260
    # проверочное значение 10 брусков - работает
    # 1:6:9


    print(" ")
    print("3.1")
    problem = pulp.LpProblem('0',pulp.LpMaximize)

    problem += x4 ,"Целевая функция"

    problem += 1*x1 + 0*x2 + 0*x3 - 1*x4 == 0,"1"
    problem += 0*x1 + 2*x2 + 1*x3 - 6*x4 == 0, "2"
    problem += 1*x1 + 0*x2 + 2*x3 - 9*x4 == 0, "3"
    problem += 1*x1 + 1*x2 + 1*x3  <= 100, "4"


    problem.solve()
    print ("Результат:")
    for variable in problem.variables():
        print (variable.name, "=", variable.varValue)
    print ("Целевая функция:")
    print (abs(value(problem.objective)))
    stop = time.time()

    print(" ")
    print("3.2")

    problem = pulp.LpProblem('0',pulp.LpMinimize)
```

```
    problem += 10*x1 + 20*x2 + 30*x3 ,"ostatok"

    problem += 1*x1 + 0*x2 + 0*x3 - 1*x4 == 0,"1"
    problem += 0*x1 + 2*x2 + 1*x3 - 6*x4 == 0, "2"
    problem += 1*x1 + 0*x2 + 2*x3 - 9*x4 == 0, "3"
    problem += 1*x1 + 1*x2 + 1*x3  == 100,"4"

    problem.solve()
    print ("Результат:")
    for variable in problem.variables():
        print (variable.name, "=", variable.varValue)
    print ("Целевая функция:")
    print (abs(value(problem.objective)))
    stop = time.time()

    #выясняем необходимое приращение
    print("3.3")
    print("3.3")
    problem = pulp.LpProblem('0',pulp.LpMaximize)
    buffer = []
    for i in range(21):
        problem = pulp.LpProblem('0', pulp.LpMaximize)
        global bugger
        problem += 1*x1 + 1*x2 + 1*x3  , "целевая функция"
        problem += 1*x1 + 0*x2 + 0*x3 - 1*x4 == 0,"1"
        problem += 0*x1 + 2*x2 + 1*x3 - 6*x4 == 0, "2"
        problem += 1*x1 + 0*x2 + 2*x3 - 9*x4 == 0, "3"
        problem += 1*x4 ==i, "4"
        problem.solve()
        buffer.append(abs(value(problem.objective)))
    print(buffer)
    # при изменении параметра колмплектов, видим линейное приращение
   полуфабрикатов. на каждую ед. комплекта требуется 6 ед. полуфабрикат
  а.
```