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: 5 (380)

Студент		Кривенкин В.П.
Преподаватель		Пономарев А.В.

Санкт-Петербург
2021

Задание на курсовую работу

Студент Кривенкин В.П.

Группа 8894

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

Исходные данные:

Текст индивидуального задания на курсовую работу в соответствии с назначенным вариантом (см. https://avponomarev.bitbucket.io/tasks/5.pdf).

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

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 05.10.2021

Дата сдачи реферата: 06.12.2021

Дата защиты реферата: 28.12.2021

Студент		Кривенкин В.П.
Преподаватель		Пономарев А.В.

Аннотация

Работа содержит детальные решения задач динамического и линейного программирования.

Первая задача (задача об инвестициях) решена с помощью уравнения Беллмана на языке программирования C++.

Вторая задача (задача о приобретении оборудования) решена с помощью пакета CVXOPT. Для этой задачи важно получить именно целочисленное решение, и в работе показано, как этого можно добиться двумя способами.

Третья задача (задача оптимального раскроя) также решена с помощью пакета CVXOPT. Это классическая задача линейного программирования, и для ее решения важно определить все осмысленные варианты разреза полуфабрикатов.

Содержание

Введение

Динамическое программирование — особый математический метод оптимизации решений, специально приспособленный к многошаговым (или многоэтапным) операциям [1]. При этом предполагается, что операция является управляемой и что управление операцией в целом складывается из элементарных (шаговых) управлений, переводящих систему из начального состояния в конечное.

Для решения задач динамического программирования строится уравнение Беллмана, после чего осуществляются обратная и прямая прогонки.

В данной работе методом динамического программирования решается задача об инвестициях, решение которой приведено на стр. 6.

Задача линейного программирования во многих случаях оказывается ассоциированной с *задачей распределительного типа*, т. е. с задачей, в которой требуется распределить ограниченные ресурсы по нескольким видам производственной деятельности [2]. Симплекс-метод, используемый для решения задач ЛП, дает ответ на вопрос, как это сделать наилучшим образом.

Если постановка задачи требует получения целочисленного решения, то может быть использован метод ветвей и границ.

К задачам линейного программирования относятся задача о приобретении оборудования и задача оптимального раскроя, решения которых приведены на стр. 12 и 21 соответственно.

1. Задача об инвестициях

Постановка задачи

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

- для 1-го цеха $f(X)=150+0.8X^{2/3}$;
- для 2-го цеха $g(Y)=110+1.6Y^{2/3}$,

где $f(X)$ - доход первого цеха за один квартал (млн. руб.), X - количество средств, вложенных за один квартал в первый цех.

Функции остатка средств за один квартал равны:

- для 1-го цеха $\varphi(X)=0.74X$;
- для 2-го цеха $\psi(X)=0.87Y$.

Количество средств, выделяемых на развитие обоих цехов в течение года, составляет 185 единиц. Средства перераспределяются поквартально и не резервируются. Требуется оптимально распределить между двумя цехами средства на планируемый год.

Формализация задачи

Итак, требуется распределить имеющиеся средства в размере $K_0=185$ (условных единиц) между цехами 1 и 2 поквартально ($m=4$) в течение года.

Будем решать задачу методом динамического программирования, по стандартной схеме.

1. Система ***S*** в данном случае — два цеха со вложенными в них средствами. Она характеризуется двумя параметрами *X* и *Y*, выражающими количества средств в цехах 1 и 2 соответственно. Этапом процесса является квартал. В процессе управления величины *X* и *Y* меняются в зависимости от двух причин:

- перераспределение средств между цехами в начале каждого квартала;
- уменьшение (трата) средств за квартал, сказывающееся в конце каждого квартала.

Управлением U_i на *i*-м шаге будут количества средств X_i и Y_i, вкладываемые в цехи 1 и 2 на этом шаге. Управление *U* операцией состоит из совокупности всех шаговых управлений:

$$U=(U_1,U_2,\dots,U_m) \quad .$$

Нам нужно найти такое (оптимальное) управление

$$u=(u_1,u_2,\dots,u_m) \quad ,$$

при котором суммарный доход, приносимый обоими цехами за *m* кварталов $W=\sum_{i=1}^{m} w_i$, был максимальным:

$$W=W_{max} \quad .$$

2. Выигрыш на *i*-м шаге:

$$w_i(K, X_i) = 150 + 0.8\,X^{2/3} + 110 + 1.6\,(K - X_i)^{2/3} \quad .$$

3. Под влиянием управления X_i (вложения средств X_i в цех 1, а $Y_i = K - X_i$ в цех 2) система на *i*-м шаге перейдет из состояния *K* в

$$K' = 0.74\,X_i + 0.87\,(K - X_i) \quad .$$

4. Основное функциональное уравнение:

$$W_i(K) = \max_{0 \le X_i \le K} \{(150 + 0.8\,X^{2/3} + 110 + 1.6\,(K - X_i)^{2/3}) + W_{i+1}(0.74\,X_i + 0.87\,(K - X_i))\} \quad .$$

Условное оптимальное управление на *i*-м шаге — то, при котором достигается этот максимум.

5. Условный оптимальный выигрыш на последнем шаге:

$$W_4(K) = \max_{0 \le X_4 \le K} \{w_4(K, X_4)\} = \max_{0 \le X_4 \le K} \{(150 + 0.8\,X_4^{2/3} + 110 + 1.6\,(K - X_4)^{2/3})\} \quad .$$

Чтобы найти этот максимум, продифференцируем выражение

$$w_4(K, X_4) = 150 + 0.8\,X_4^{2/3} + 110 + 1.6\,(K - X_4)^{2/3}$$

при фиксированном *K* по X_4 и приравняем производную нулю

$$\frac{\partial w_4}{\partial X_4} = 0.8 \cdot 0.66 \cdot X_4^{-1/3} - 1.6 \cdot 0.66 \cdot (K - X_4)^{-1/3} = 0 \quad ;$$

$$\frac{0.53}{\sqrt[3]{X_4}} - \frac{1.06}{\sqrt[3]{K - X_4}} = 0 \quad ;$$

$$\sqrt[3]{K - X_4} = 2\sqrt[3]{X_4};\ X_4 \neq 0,\ K - X_4 \neq 0 \quad ;$$

$$X_4 = K/9 \quad .$$

Таким образом, условно оптимальное управление на последнем (четвертом) шаге найдено: $X_4 = K/9$ и, соответственно, $Y_4 = 8K/9$. На дальнейших шагах задача решается численно (графически).

Графики зависимости выигрыша от управления при различных состояниях

Рассмотрим последний этап, оптимальное управление X_4 на котором нам уже известно. Имеющиеся на этом этапе средства K находятся в диапазоне от $K_{min}=0.74^3\cdot 185=74.97$ до $K_{max}=0.87^3\cdot 185=121.82$. Возьмем эти значения K и еще несколько промежуточных и построим график зависимости выигрыша от управления при этих состояниях (рис. 1).

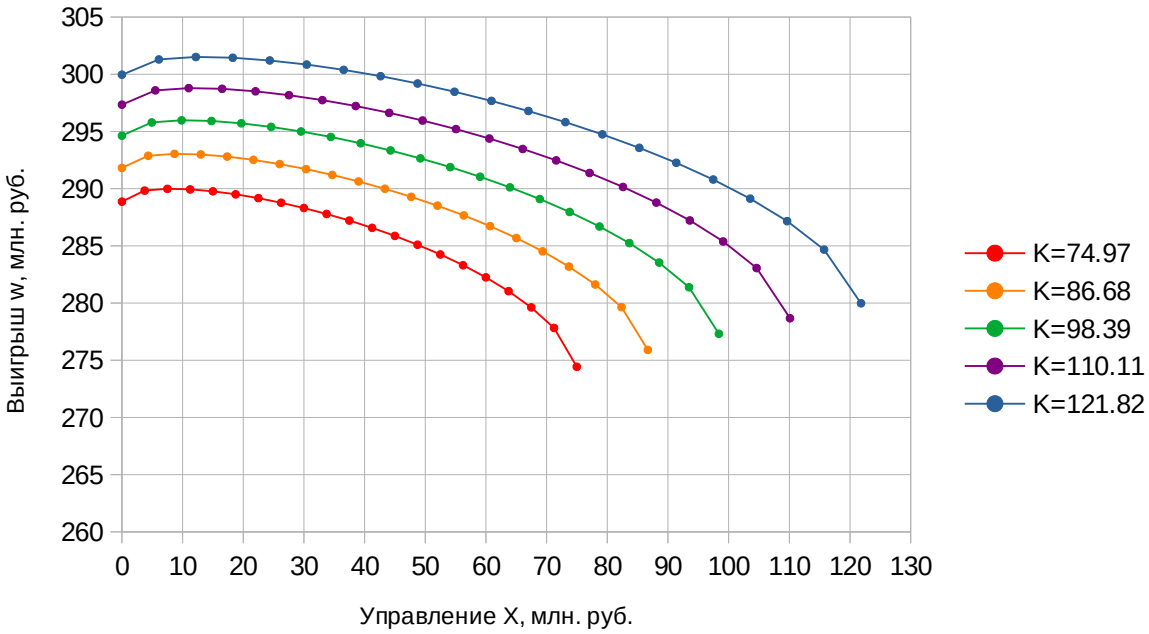

Рис. 1. График зависимости выигрыша от управления на последнем этапе.

Присмотревшись, мы действительно видим, что наибольший выигрыш достигается при управлении $X_4=K/9$.

Этот график поможет нам при построении схожего графика для предпоследнего этапа, к которому мы и переходим.

Рассмотрим предпоследний этап, оптимальное управление X_3 на котором нам предстоит определить. По аналогии: K находится в диапазоне от $K_{min}=0.74^2\cdot 185=101.31$ до $K_{max}=0.87^2\cdot 185=140.03$. Дополнительно берем еще несколько значений K и строим график зависимости выигрыша от управления при этих состояниях (рис. 2).

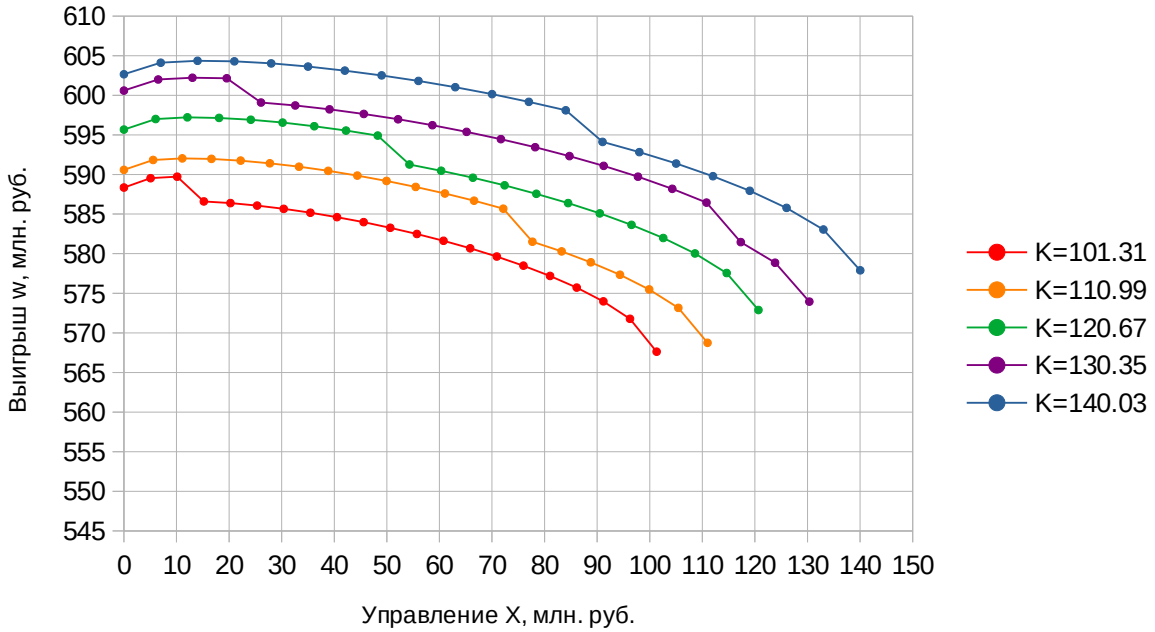

Рис. 2. График зависимости выигрыша от управления на предпоследнем этапе.

Для состояния K_{min} условный максимальный доход на двух последних шагах достигается при $X_3=10.13$, а для K_{max} — при $X_3=14.00$.

На графике видны характерные «ступени», означающие, что при текущем управлении уже не удастся получить такой же выигрыш на следующем этапе, как при предыдущем (несколько меньшем) управлении.

Для последних двух этапов графики зависимости выигрыша от управления строятся аналогично.

Фрагменты таблиц с расчетами

В программе предусмотрена возможность вывода таблиц с промежуточными вычислениями. Для каждого этапа (квартала) выводится отдельная таблица. На рис. 3 продемонстрирован вывод программой таблиц при следующих параметрах: $N=4$ (дискретизация по K) и $M=20$ (дискретизация по X).

```
valery@dave:~/Desktop/dm/src/course/investment$ ./a.out
Table 1 (K, w, X)
  0.00,   0.00,   0.00
  0.00,   0.00,   0.00
  0.00,   0.00,   0.00
  0.00,   0.00,   0.00
185.00, 1225.88,  18.50

Table 2 (K, w, X)
136.90, 896.91,  13.69
142.91, 903.40,  14.29
148.92, 904.69,  14.89
154.94, 910.97,  15.49
160.95, 912.23,  16.09

Table 3 (K, w, X)
101.31, 589.72,  10.13
110.99, 592.03,  11.10
120.67, 597.21,  12.07
130.35, 602.21,  13.03
140.03, 604.35,  14.00

Table 4 (K, w, X)
 74.97, 289.98,   8.33
 86.68, 293.04,   9.63
 98.39, 295.97,  10.93
110.11, 298.79,  12.23
121.82, 301.51,  13.54
```

Рис. 3. Демонстрация вывода программой таблиц.

Результат

Результат и время работы программы при параметрах $N = M = 1000$ показан на рис. 4.

```
valery@dave:~/Desktop/dm/src/course/investment$ time ./a.out
Total income: 1229.97
        Stage 1: X=  8.32, Y=176.68
        Stage 2: X=  9.43, Y=150.44
        Stage 3: X= 10.34, Y=127.52
        Stage 4: X= 13.18, Y=105.41

real    0m1.154s
user    0m1.154s
sys     0m0.000s
```

Рис. 4. Результат и время работы программы.

Мы видим, что программа была выполнена за время $t \approx 1.2\,c$, а имеющиеся средства следует распределить следующим образом (табл. 1).

Таблица 1. Оптимальная стратегия инвестирования.

	Квартал 1	Квартал 2	Квартал 3	Квартал 4
X, млн. руб.	8.32	9.43	10.34	13.18
Y, млн. руб	176.68	150.44	127.52	105.41

Если мы будем придерживаться этой стратегии инвестирования, то суммарный доход, который мы получим в конце года, будет равен $W = W_{max} = 1229.97\,млн.\,руб.$

2. Задача о приобретении оборудования

Постановка задачи

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 80%. 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	34	26	1000
Тип 2	57	19	2100
Тип 3	32	28	900
Тип 4	59	22	1800
Тип 5	53	24	2000

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, связанного 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.

Формализация задачи

Переменные

x_j — количество приобретаемого оборудования j-го типа; $j=1,2,\dots,5$.

Ресурсы

Денежные средства: $R=0.8\cdot 1299.97=983\,млн.руб$; площадь: $S=290\,кв.м$.

Математическая модель

$$1000x_1+2100x_2+900x_3+1800x_4+2000x_5\to max$$
$$34x_1+57x_2+32x_3+59x_4+53x_5\le 983 \quad ,(1)$$
$$26x_1+19x_2+28x_3+22x_4+24x_5\le 290 \quad ,(2)$$
$$x_j\ge 0 \;;\; x_j-целое \;;\; j=1,2,\dots,5 \quad .$$

Решение методом ветвей и границ

Этот метод получения оптимального целочисленного решения задач ЛП требует сначала решения исходной задачи с ослабленными ограничениями (без требования целочисленности), затем решения порожденных ею задач с дополнительными ограничениями на переменные.

Процесс поиска решения может быть представлен в виде дерева (см. рис. 3 и 4). Каких-либо рекомендаций по наилучшему направлению поиска, к сожалению, не существует.

Подготовка матриц

Подготовим математическую модель исходной задачи с ослабленными ограничениями для решения с помощью пакета CVXOPT (таблица 1).

Таблица 1. Подготовка матриц

	x1	**x2**	**x3**	**x4**	**x5**	**Нер-во**	**Пр. часть**
c	1000	2100	900	1800	2000	-	max **(-1)**
y1	34	57	32	59	53	<=	983
y2	26	19	28	22	24	<=	290
y3	1	0	0	0	0	>= **(-1)**	0
y4	0	1	0	0	0	>= **(-1)**	0
y5	0	0	1	0	0	>= **(-1)**	0
y6	0	0	0	1	0	>= **(-1)**	0
y7	0	0	0	0	1	>= **(-1)**	0

Заполнение матриц и решение задачи

Заполним матрицы исходной задачи (рис. 1).

```
from cvxopt import matrix, solvers

c = matrix([-1000, -2100, -900, -1800, -2000], tc='d')

G = matrix([[34, 57, 32, 59, 53],
            [26, 19, 28, 22, 24],
            [-1,  0,  0,  0,  0],
            [ 0, -1,  0,  0,  0],
            [ 0,  0, -1,  0,  0],
            [ 0,  0,  0, -1,  0],
            [ 0,  0,  0,  0, -1]], tc='d')

h = matrix([983, 290, 0, 0, 0, 0, 0], tc='d')
```

Рис. 1. Матрицы задачи

Получим решение исходной задачи (рис. 2).

```
print('Status', solution['status'])
print('x = \n', solution['x'])
print('Objective: ', -solution['primal objective'])
```

```
Status optimal
x =
 [ 0.00e+00]
[ 1.53e+01]
[ 0.00e+00]
[ 0.00e+00]
[ 0.00e+00]

Objective:  32052.63157894737
```

Рис. 2. Решение задачи

Видим, что полученное решение является оптимальным, но не является целочисленным ($x_2^*=15.3$), поэтому нужно вводить дополнительные ограничения и решать подзадачи. Процесс поиска наилучшего целочисленного решения методом ветвей и границ представлен на рис. 3 и 4.

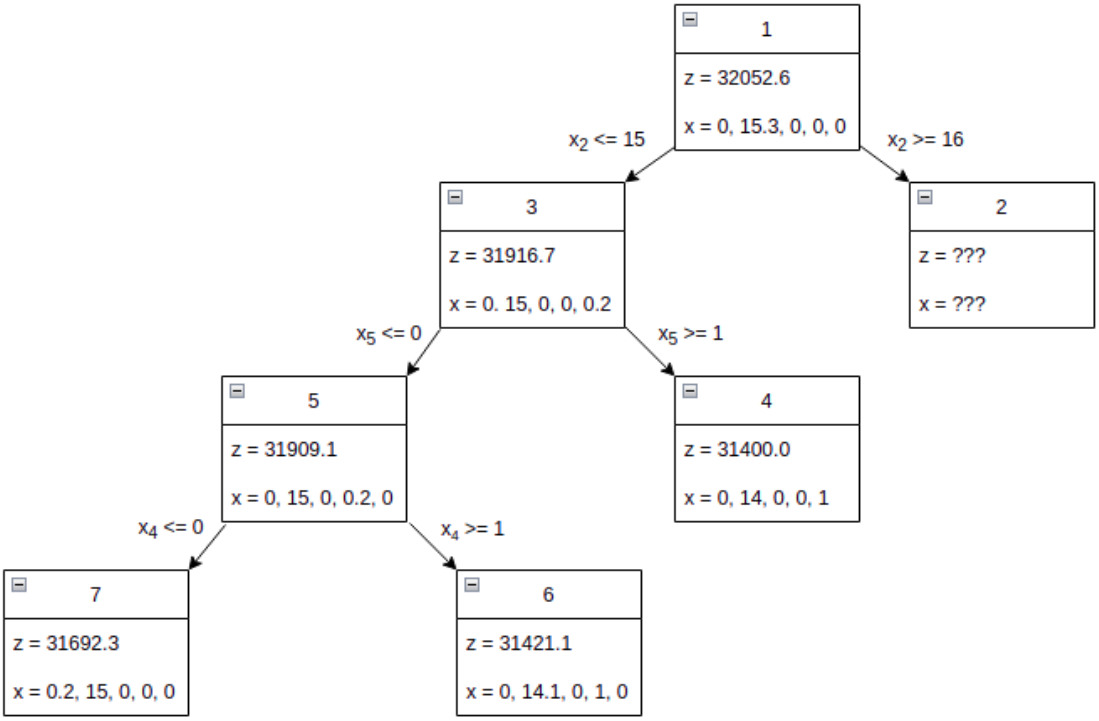

Рис. 3. Поиск наилучшего целочисленного решения

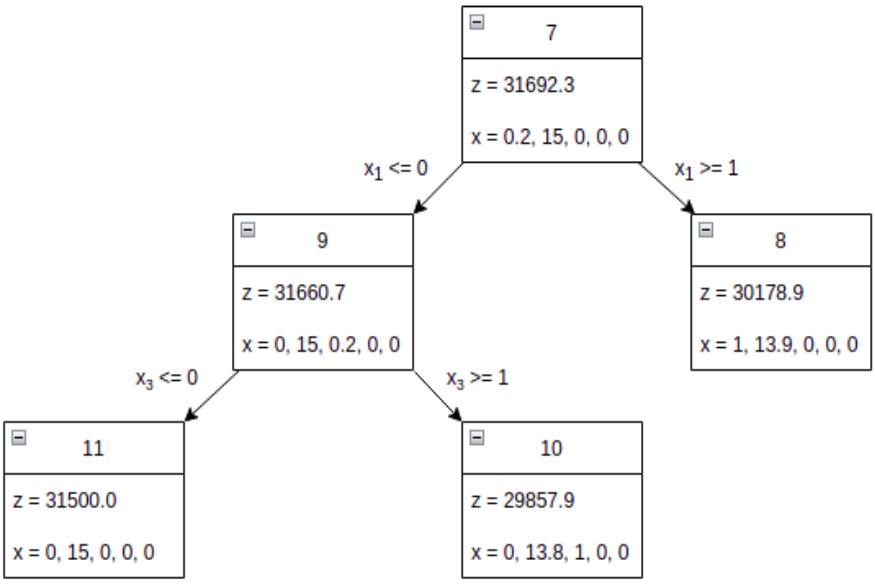

Рис. 4. Поиск наилучшего целочисленного решения (продолжение)

Итак, поиск по задачам **1-2-...-11** приводит нас к наилучшему целочисленному решению $x=[0,15,0,0,0]$, обеспечивающему производительность $z=31500.0\,ш т.$. Это решение является довольно очевидным, однако, чтобы его получить, нам пришлось наложить ограничения равенства нулю на все переменные, кроме x_2.

Отметим несколько важных моментов.

- Для подзадачи 2 мы не смогли получить никакого решения из-за нарушения ограничения по площади ($16 ед. \cdot 19 кв.м > 290 кв.м$).
- Решив подзадачу 4, мы получили нижнюю границу целочисленного решения ($z = 31400 шт.$).
- Рассмотрение подзадач ветви 6 мы прекратили, так как разница между значением функции цели z и нижней границей ($\Delta z = 21.1$) была слишком мала (коэффициенты целевой функции не могут обеспечить целочисленного решения с ненулевым числом десятков и/или единиц).
- Подзадачи ветвей 8 и 10 мы не стали рассматривать, потому что нижняя целочисленная граница превышает значения их целевых функций z.

Проверка решения

Проверим, что полученное нами решение является верным. Для этого решим задачу, используя функцию ilp для решения задач целочисленного и смешанного программирования из солвера glpk, и сравним результаты (рис. 5).

```
from cvxopt import matrix, glpk

c = matrix([-1000, -2100, -900, -1800, -2000], tc='d')

G = matrix([[34, 57, 32, 59, 53],
            [26, 19, 28, 22, 24],
            [-1,  0,  0,  0,  0],
            [ 0, -1,  0,  0,  0],
            [ 0,  0, -1,  0,  0],
            [ 0,  0,  0, -1,  0],
            [ 0,  0,  0,  0, -1]], tc='d')

h = matrix([983, 290, 0, 0, 0, 0, 0], tc='d')
(status, x) = glpk.ilp(c, G.T, h, I=set(range(5)))

print('Status: ', status)
print('x = \n', x)
```

```
Status:  optimal
x =
 [ 0.00e+00]
[ 1.50e+01]
[ 0.00e+00]
[ 0.00e+00]
[ 0.00e+00]
```

Рис. 5. Решение с помощью функции ilp

Получаем то же решение, что и методом ветвей и границ.

Анализ чувствительности

Проведем анализ чувствительности полученного целочисленного решения к определенным изменениям исходной модели.

Анализ правых частей

Определим, какие ресурсы являются дефицитными, а какие — нет (рис. 6).

```
R = 983
S = 290

h = matrix([R, S, 0, 0, 0, 0, 0], tc='d')
(status, x) = glpk.ilp(c, G.T, h, I=set(range(5)))

# print('Status: ', status)
print('x = \n', x)

print('Free money: ', R - x[0]*34 - x[1]*57 - x[2]*32 - x[3]*59 - x[4]*53)
print('Free space: ', S - x[0]*26 - x[1]*19 - x[2]*28 - x[3]*22 - x[4]*24)
```

```
x =
 [ 0.00e+00]
[ 1.50e+01]
[ 0.00e+00]
[ 0.00e+00]
[ 0.00e+00]

Free money:  128.0
Free space:  5.0
```

Рис. 6. Определение дефицитности ресурсов

Мы видим, что денежные средства R — *недефицитный* ресурс, а площадь S — *дефицитный* ресурс.

Это, во-первых, означает, что решение не изменится, если 128 млн. денежных средств будут потрачены на что-то другое (рис. 7).

```
R = 983 - 128
S = 290

h = matrix([R, S, 0, 0, 0, 0, 0], tc='d')
(status, x) = glpk.ilp(c, G.T, h, I=set(range(5)))

# print('Status: ', status)
print('x = \n', x)

print('Free money: ', R - x[0]*34 - x[1]*57 - x[2]*32 - x[3]*59 - x[4]*53)
print('Free space: ', S - x[0]*26 - x[1]*19 - x[2]*28 - x[3]*22 - x[4]*24)
```

```
x =
 [ 0.00e+00]
[ 1.50e+01]
[ 0.00e+00]
[ 0.00e+00]
[ 0.00e+00]

Free money:  0.0
Free space:  5.0
```

Рис. 8. Альтернативное применение денежных средств

Во-вторых, что, увеличив площадь на $2 \cdot 19 - 5 = 33 кв.м$, мы сможем приобрести еще две единицы оборудования второго типа, увеличив производительность цеха на $2 \cdot 2100 = 4200 шт.$ (рис. 9).

```
13 R = 983
14 S = 290 + 33
15
16 h = matrix([R, S, 0, 0, 0, 0, 0], tc='d')
17 (status, x) = glpk.ilp(c, G.T, h, I=set(range(5)))
18
19 # print('Status: ', status)
20 print('x = \n', x)
21
22 print('Free money: ', R - x[0]*34 - x[1]*57 - x[2]*32 - x[3]*59 - x[4]*53)
23 print('Free space: ', S - x[0]*26 - x[1]*19 - x[2]*28 - x[3]*22 - x[4]*24)
```

```
x =
 [ 0.00e+00]
[ 1.70e+01]
[ 0.00e+00]
[ 0.00e+00]
[ 0.00e+00]

Free money:  14.0
Free space:  0.0
```

Рис. 9. Эффект от дополнительной площади

Отметим, что в этом случае денежные средства также становятся *дефицитным* ресурсом.

Вопрос о выборе двух и более типов оборудования

При фиксированной площади $S = 290\,кв.м$ нас интересует диапазон денежных средств, в пределах которого оптимальным является выбор двух и более типов оборудования.

Покажем, что такой диапазон не является единственным (рис. 10).

```
13 R = 100
14 S = 290
15
16 while R <= 1000:
17     h = matrix([R, S, 0, 0, 0, 0, 0], tc='d')
18     (status, xs) = glpk.ilp(c, G.T, h, I=set(range(5)))
19     n_types = len([x for x in xs if x > 0])
20     print(f"R = {R}, num of types = {n_types}")
21     R += 25
22
R = 350, num of types = 2
R = 375, num of types = 2
R = 400, num of types = 1
R = 425, num of types = 1
R = 450, num of types = 2
R = 475, num of types = 3
R = 500, num of types = 2
R = 525, num of types = 3
R = 550, num of types = 2
R = 575, num of types = 3
R = 600, num of types = 2
R = 625, num of types = 3
R = 650, num of types = 2
R = 675, num of types = 2
R = 700, num of types = 3
R = 725, num of types = 2
R = 750, num of types = 1
R = 775, num of types = 2
R = 800, num of types = 1
```

Рис. 10. Диапазон денежных средств (шаг - 25 млн. руб)

Видим, что переход от решения о приобретении двух и более видов оборудования к решению о приобретении одного вида оборудования при $R_{min} = 100\,млн.руб$, $R_{max} = 1000\,млн.руб$, $\Delta R = 50\,млн.руб$ происходит по крайней мере дважды.

Уменьшим величину шага до 1 млн. руб и найдем границы *ближайшего* к $R = 983\ млн.\ руб$ диапазона денежных средств, в пределах которого оптимальна закупка двух и более типов оборудования (рис. 11).

```
13 R = 800
14 S = 290
15
16 while R <= 983:
17     h = matrix([R, S, 0, 0, 0, 0, 0], tc='d')
18     (status, xs) = glpk.ilp(c, G.T, h, I=set(range(5)))
19     n_types = len([x for x in xs if x > 0])
20     print(f"R = {R}, num of types = {n_types}")
21     R += 1
22
```

```
R = 846, num of types = 1
R = 847, num of types = 1
R = 848, num of types = 1
R = 849, num of types = 1
R = 850, num of types = 1
R = 851, num of types = 2
R = 852, num of types = 2
R = 853, num of types = 2
R = 854, num of types = 2
R = 855, num of types = 1
R = 856, num of types = 1
```

Рис. 11. Ближайший к R = 290 млн. руб диапазон

Таким образом, границы искомого диапазона

$$R_{min} = 851\ млн.\ руб, R_{max} = 854\ млн.\ руб\ .$$

Пусть $R = 853\ млн.\ руб$. Посмотрим на оптимальное решение, чтобы увидеть, в пользу каких двух видов оборудования делается выбор (рис. 12).

```
13 R = 853
14 S = 290
15
16 h = matrix([R, S, 0, 0, 0, 0, 0], tc='d')
17 (status, x) = glpk.ilp(c, G.T, h, I=set(range(5)))
18
19 print('Status: ', status)
20 print('x = \n', x)
```

```
Status:  optimal
x =
 [ 0.00e+00]
[ 1.40e+01]
[ 0.00e+00]
[ 0.00e+00]
[ 1.00e+00]
```

Рис. 12. Целочисленный результат для R = 853 млн. руб.

На этом мы завершаем анализ чувствительности.

3. Задача оптимального раскроя

Постановка задачи

Выпускаемая в 3-м цехе продукция, представляющая собой полуфабрикат определенного типоразмера постоянного сечения и длиной 450 см, разрезается на заготовки длиной 450 см, 230 см, 90 см в комплектности, определяемой соотношением 2:3:4.

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).

Формализация задачи 1

Переменные

Пусть n — число комплектов заготовок. Прежде чем строить математическую модель, укажем все осмысленные варианты разреза полуфабриката на заготовки (табл. 1).

Таблица 1. Варианты раскроя

Длина заготовки, см	Вариант разреза заготовки			Количество заготовок, шт.
	1	2	3	
450	1	0	0	$2n$
230	0	1	0	$3n$
90	0	2	5	$4n$

Теперь мы можем ввести переменные следующим образом: x_j — количество полуфабрикатов, разрезанных по варианту j, $j = 1, 2, 3$.

Отметим, что в этой постановке задачи про потери материала мы не думаем.

Математическая модель

$$n \to max$$

$$x_1 = 2n \quad , (1)$$

$$x_2 = 3n \quad , (2)$$

$$2x_2 + 5x_3 \geq 4n \quad , (3)$$

$$n \geq 0, x_j \geq 0, j = 1, 2, 3 \quad . (4)$$

Очевидно, что третий вариант раскроя никогда не будет использоваться, поэтому ограничение (3) вырождается в $6n \geq 4n$ и может быть вычеркнуто.

Решение задачи 1

Подготовим математическую модель для решения задачи с помощью пакета CVXOPT (табл. 2).

Таблица 2. Подготовка матриц

	x1	x2	x3	n	Знак	Пр. часть
c	0	0	0	1	-	max **(-1)**
y1	1	0	0	-2	=	0
y2	0	1	0	-3	=	0
y3	1	0	0	0	>= **(-1)**	0
y4	0	1	0	0	>= **(-1)**	0
y5	0	0	1	0	>= **(-1)**	0
y6	0	0	0	1	>= **(-1)**	0
y7	1	1	1	0	<=	100

Ограничение y7 устанавливает верхнюю границу числа полуфабрикатов.

```
from cvxopt import matrix, solvers

c = matrix([0, 0, 0, -1], tc='d')

G = matrix([[-1,  0,  0,  0],
            [ 0, -1,  0,  0],
            [ 0,  0, -1,  0],
            [ 0,  0,  0, -1],
            [ 1,  1,  1,  0]], tc='d')

h = matrix([0, 0, 0, 0, 100], tc='d')

A = matrix([[ 1,  0,  0, -2],
            [ 0,  1,  0, -3]], tc='d')

b = matrix([0, 0], tc='d')

solution = solvers.lp(c, G.T, h, A.T, b, solver='glpk')
print('Status', solution['status'])
print('x = \n', solution['x'])
print('Objective:', -solution['primal objective'])
```

```
Status optimal
x =
 [ 4.00e+01]
[ 6.00e+01]
[ 0.00e+00]
[ 2.00e+01]

Objective: 20.0
```

Рис. 1. Решение первой задачи

На рис. 1 приведено решение первой задачи. Результат: разрезав 40 полуфабрикатов первым способом и 60 — вторым, получим 20 комплектов.

Теперь рассмотрим эту же задачу с точки зрения минимизации потерь материала.

Формализация задачи 2

Переменные

В этой постановке задачи мы будем минимизировать потери, которые связаны не только с обрезками материала, но и с избыточным числом заготовок.

Добавим строку потерь материала в таблицу вариантов раскроя (табл. 3).

Таблица 3. Раскрой с учетом потерь

Длина заготовки, см	Вариант разреза заготовки			Количество заготовок, шт.
	1	2	3	
450	1	0	0	2*n*
230	0	1	0	3*n*
90	0	2	5	4*n*
Потери, см	0	40	0	

Введем переменные: x_j — количество полуфабрикатов, разрезанных по варианту j, $j = 1, 2, 3$; y_i — избыточное число заготовок типа i, $i = 1, 2, 3$.

Математическая модель

$$40\,x_2 + 450\,y_1 + 230\,y_2 + 90\,y_3 \to min$$

$$x_1 - y_1 = 40 \quad , (1)$$

$$x_2 - y_2 = 60 \quad , (2)$$

$$2\,x_2 + 5\,x_3 - y_3 = 80 \quad , (3)$$

$$x_j \geq 0, j = 1,2,3; y_i \geq 0, i = 1,2,3 \quad . (4)$$

Отметим, что правые части ограничений 1, 2, 3 такие, потому что мы хотим получить 20 комплектов заготовок. Так мы сможем сравнить решения задачи в первой и второй постановках.

Решение задачи 2

Подготовим математическую модель для решения задачи с помощью пакета CVXOPT (табл. 4).

Таблица 4. Подготовка матриц

	x1	x2	x3	y1	y2	y3	Знак	Пр. часть
c	0	40	0	450	230	90	-	min
y1	1	0	0	-1	0	0	=	40
y2	0	1	0	0	-1	0	=	60
y3	0	2	5	0	0	-1	=	80
y4	1	0	0	0	0	0	>= **(-1)**	0
y5	0	1	0	0	0	0	>= **(-1)**	0
y6	0	0	1	0	0	0	>= **(-1)**	0
y7	0	0	0	1	0	0	>= **(-1)**	0
y8	0	0	0	0	1	0	>= **(-1)**	0
y9	0	0	0	0	0	1	>= **(-1)**	0

```
from cvxopt import matrix, solvers

c = matrix([0, 40, 0, 450, 230, 90], tc='d')

G = matrix([[-1,  0,  0,  0,  0,  0],
            [ 0, -1,  0,  0,  0,  0],
            [ 0,  0, -1,  0,  0,  0],
            [ 0,  0,  0, -1,  0,  0],
            [ 0,  0,  0,  0, -1,  0],
            [ 0,  0,  0,  0,  0, -1]], tc='d')

h = matrix([0, 0, 0, 0, 0, 0], tc='d')
A = matrix([[ 1,  0,  0, -1,  0,  0],
            [ 0,  1,  0,  0, -1,  0],
            [ 0,  2,  5,  0,  0, -1]], tc='d')

b = matrix([40, 60, 80], tc='d')
solution = solvers.lp(c, G.T, h, A.T, b, solver='glpk')
print('Status', solution['status'])
print('x = \n', solution['x'])
print('Objective:', solution['primal objective'])
```

```
Status optimal
x =
 [ 4.00e+01]
[ 6.00e+01]
[ 0.00e+00]
[ 0.00e+00]
[ 0.00e+00]
[ 4.00e+01]

Objective: 6000.0
```

Рис. 2. Решение второй задачи

На рис. 2 приведено решение второй задачи. Результат тот же, только теперь мы видим общие потери материала, которые составляют $40 \cdot 60 + 40 \cdot 90 = 6000$ см.

Параметрическое исследование правых частей

Посмотрим на вектор теневых цен первой задачи (рис. 3).

```
23 print('z = \n', solution['z'])
```

```
z =
 [-0.00e+00]
[-0.00e+00]
[ 2.00e-01]
[-0.00e+00]
[ 2.00e-01]
```

Рис. 3. Теневые цены

Видим, что, увеличив правую часть ограничения y7 на единицу, значение целевой функции вырастет на 0.2. Следовательно, чтобы произвести дополнительный комплект заготовок, мы должны увеличить число полуфабрикатов на 5. Убедимся, что это так (рис. 4).

```
11 h = matrix([0, 0, 0, 0, 105], tc='d')
12
13 A = matrix([[ 1,  0,  0, -2],
14             [ 0,  1,  0, -3]], tc='d')
15
16 b = matrix([0, 0], tc='d')
17
18 solution = solvers.lp(c, G.T, h, A.T, b, solver='glpk')
19 print('Status', solution['status'])
20 print('x = \n', solution['x'])
21 print('Objective:', -solution['primal objective'])
```

```
Status optimal
x =
 [ 4.20e+01]
[ 6.30e+01]
[ 0.00e+00]
[ 2.10e+01]

Objective: 21.0
```

Рис. 4. Увеличение числа комплектов на единицу (задача 1)

Убедимся, что задача во второй постановке дает то тоже решение, задав число комплектов заготовок равным 21 (рис. 5).

```
17 b = matrix([42, 63, 84], tc='d')
18 solution = solvers.lp(c, G.T, h, A.T, b, solver='glpk')
19 print('Status', solution['status'])
20 print('x = \n', solution['x'])
21 print('Objective:', solution['primal objective'])
```

```
Status optimal
x =
 [ 4.20e+01]
[ 6.30e+01]
[-2.84e-15]
[ 0.00e+00]
[ 0.00e+00]
[ 4.20e+01]

Objective: 6300.0
```

Рис. 5. Увеличение числа комплектов на единицу (задача 2)

Получаем то же решение. При этом видим, что потери материала увеличились на $40 \cdot 3 + 2 \cdot 90 = 300$ см.

Проводить такое же исследование для другого числа полуфабрикатов не имеет смысла, поскольку, как было показано выше, выбор всегда делается в пользу первых двух вариантов раскроя. Таким образом, увеличение числа полуфабрикатов на 5 единиц всегда будет давать один дополнительный комплект заготовок.

Заключение

Итак, при выполнении данной работы были решены три задачи динамического и линейного программирования.

Для задачи динамического программирования (задачи об инвестициях) так называемый «солвер» был разработан с нуля на языке C++. Дело в том, что задачи, которые могут быть решены методом ДП весьма разнообразны, и какой-либо канонической формы задач ДП не существует. Поэтому, как правило, «солвер» разрабатывается под конкретную задачу с учетом принципов ДП: наличия нескольких этапов, решения от конца к началу, поиска условно-оптимальных управлений и т. д.

С задачами линейного программирования дела обстоят несколько проще. Любая задача ЛП может быть приведена к канонической форме и решена с помощью имеющихся инструментов. Основная сложность — составление математической модели задачи.

Если в задаче есть ограничение целочисленности, то часто используется метод ветвей и границ, последовательно добавляющий новые ограничения к исходной задаче ЛП с ослабленными ограничениями.

К задачам ЛП относятся задача о приобретении оборудования и задача оптимального раскроя, причем первая имеет ограничение целочисленности.

Обе задачи были решены нами с помощью пакета CVXOPT, который позволяет решать множество задач выпуклой оптимизации, в том числе и задачи целочисленного и линейного программирования.

Список источников

[1] Вентцель Е. С. *Исследование операций*. М., «Советское радио», 1972.

[2] Таха Х. *Введение в исследование операций: В 2-х книгах. Кн. 1*. М.: Мир, 1985.

[3] Пономарев А. В. *Решение задач линейного программирования с использованием GNU Octave, GLPK и Python.*

Приложение А

Исходный код программ

Исходный код программы, решающей задачу об инвестициях, доступен по ссылке: https://github.com/valery42/Decision-Theory/blob/main/src/course/investment/investment.cpp

Блокноты программы, решающей задачу о приобретении оборудования, доступны по ссылке:
https://github.com/valery42/Decision-Theory/tree/main/src/course/equipment

Блокноты программы, решающей задачу оптимального раскроя, доступны по ссылке:
https://github.com/valery42/Decision-Theory/tree/main/src/course/cutting