

Анализ чувствительности в задачах ЛП с помощью Python

(а также GNU Octave и GLPK/GMPL)

Цель

- Анализ чувствительности позволяет получить ответ на вопрос как изменится найденное оптимальное решение задачи ЛП, при *незначительном* изменении исходных данных. Как правило, интересуется следующее:
 1. Как изменится решение, если изменится количество ресурса (значения из вектора правых частей b_i)?
 2. Как изменится решение задачи, если изменятся параметры целевой функции c_i ?
- Ответ на первый вопрос связан с понятием теневой цены ресурса (*shadow price*), а на второй – с понятием приведенной цены (нормированной стоимости, уменьшенной стоимости, *reduced cost*).
- В качестве примера рассмотрим достаточно типичную задачу линейного программирования производственного типа. Это даст возможность раскрыть экономический смысл понятий теневой и приведенной цены.

Постановка задачи (1/2)

- Фабрика производит три вида продукции: П1, П2 и П3.
- Известна цена на продукцию для распространителей и приблизительный спрос на каждый из видов продукции в неделю (см. Таблицу 1).
- Процессы производства продукции разных видов имеют отличия. На фабрике есть три цеха: Ц1, Ц2 и Ц3. Для производства продукции П1 необходимы только технологические операции, производимые цехом Ц1, для П2 – Ц1 и Ц3, для производства П3 – необходима полная технологическая цепочка, включающая обработку во всех трех цехах.
- В цехах Ц1 и Ц2 продукция разных видов обрабатывается одинаково, и известна общая производительность этих цехов в единицах обработанной продукции в неделю, то в цехе Ц3 предполагается ручная обработка (см. Таблицу 2).
- Из всех видов материалов, используемых при производстве продукции, ограниченным является только один, поставки его в неделю и потребности для каждого из видов продукции приведены в таблице 3.
- Необходимо составить производственный план на неделю, максимизирующий выручку от реализации продукции.

Постановка задачи (2/2)

Таблица 1 – Характеристики продукции

Вид продукции	Цена, руб.	Спрос, шт. в неделю
П1	1200	35
П2	2500	25
П3	1400	30

Таблица 2 – Производительность цехов

Ц1, шт. в неделю	Ц2, шт. в неделю	Ц3, часов (П2/П3/Общий фонд)
40	20	8/2/80

Таблица 3 – Материалы

Поставки в неделю, кг	Потребление на ед. продукта П1, кг	Потребление на ед. продукта П2, кг	Потребление на ед. продукта П3, кг
50	0,8	0,6	0,7

Формальная постановка задачи

- Переменные

- $x_i, i \in \{1..3\}$ – количество единиц продукции i -того вида, которое необходимо произвести за неделю [шт].

- Тогда

$$1200x_1 + 2500x_2 + 1400x_3 \rightarrow \max$$

$$x_1 + x_2 + x_3 \leq 40 \quad (1)$$

$$x_3 \leq 20 \quad (2)$$

$$8x_2 + 2x_3 \leq 80 \quad (3)$$

$$x_1 \leq 35 \quad (4)$$

$$x_2 \leq 25 \quad (5)$$

$$x_3 \leq 30 \quad (6)$$

$$0,8x_1 + 0,6x_2 + 0,7x_3 \leq 50 \quad (7)$$

$$x_{1,2,3} \geq 0$$

Таблица

	x1	x2	x3	Неравенство	Пр. часть
с	1200	2500	1400	-	max (-1)
y1	1	1	1	<=	40
y2	0	0	1	<=	20
y3	0	8	2	<=	80
y4	1	0	0	<=	35
y5	0	1	0	<=	25
y6	0	0	1	<=	30
y7	0,8	0,6	0,7	<=	50

Python: CVXOPT

`cvxopt.solvers.lp(c, G, h[, A, b[, solver[, primalstart[, dualstart]]]])`

$$\begin{array}{ll}\text{minimize} & c^T x \\ \text{subject to} & Gx + s = h \\ & Ax = b \\ & s \succeq 0\end{array}$$

$$\begin{array}{ll}\text{maximize} & -h^T z - b^T y \\ \text{subject to} & G^T z + A^T y + c = 0 \\ & z \succeq 0.\end{array}$$

Таким образом:

- c – коэффициенты переменных в целевой функции;
- G – коэффициенты переменных в ограничениях-неравенствах (\leq);
- h – правые части ограничений-неравенств (\leq);
- A – коэффициенты переменных в ограничениях-равенствах;
- b – правые части ограничений-равенств.

CVXOPT: Подготовка матриц

	x1	x2	x3	Неравенство	Пр. часть
c	1200	2500	1400	-	max (-1)
y1	1	1	1	<=	40
y2	0	0	1	<=	20
y3	0	8	2	<=	80
y4	1	0	0	<=	35
y5	0	1	0	<=	25
y6	0	0	1	<=	30
y7	0,8	0,6	0,7	<=	50
y8	1	0	0	>=	0
y9	0	1	0	>=	0
y10	0	0	1	>=	0

```
from cvxopt import matrix, solvers
```

```
c = matrix([-1200, -2500, -1400], tc='d')
```


CVXOPT: Подготовка матриц

	x1	x2	x3	Неравенство	Пр. часть
c	1200	2500	1400	-	max (-1)
y1	1	1	1	<=	40
y2	0	0	1	<=	20
y3	0	8	2	<=	80
y4	1	0	0	<=	35
y5	0	1	0	<=	25
y6	0	0	1	<=	30
y7	0,8	0,6	0,7	<=	50
y8	1	0	0	>= (-1)	0
y9	0	1	0	>= (-1)	0
y10	0	0	1	>= (-1)	0

```
from cvxopt import matrix, solvers
```

```
c = matrix([-1200, -2500, -1400], tc='d')
```

```
G = matrix([[ 1, 1, 1],  
            [ 0, 0, 1],  
            [ 0, 8, 2],  
            [ 1, 0, 0],  
            [ 0, 1, 0],  
            [ 0, 0, 1],  
            [0.8, 0.6, 0.7],  
            [-1, 0, 0],  
            [ 0, -1, 0],  
            [ 0, 0, -1]], tc='d')
```

CVXOPT: Подготовка матриц

	x1	x2	x3	Неравенство	Пр. часть
c	1200	2500	1400	-	max (-1)
y1	1	1	1	<=	40
y2	0	0	1	<=	20
y3	0	8	2	<=	80
y4	1	0	0	<=	35
y5	0	1	0	<=	25
y6	0	0	1	<=	30
y7	0,8	0,6	0,7	<=	50
y8	1	0	0	>= (-1)	0
y9	0	1	0	>= (-1)	0
y10	0	0	1	>= (-1)	0

```
from cvxopt import matrix, solvers
```

```
c = matrix([-1200, -2500, -1400], tc='d')
```

```
G = matrix([[ 1, 1, 1],  
            [ 0, 0, 1],  
            [ 0, 8, 2],  
            [ 1, 0, 0],  
            [ 0, 1, 0],  
            [ 0, 0, 1],  
            [0.8, 0.6, 0.7],  
            [-1, 0, 0],  
            [ 0, -1, 0],  
            [ 0, 0, -1]], tc='d')
```

```
h = matrix([40, 20, 80, 35, 25, 30,  
           50, 0, 0, 0], tc='d')
```

Решение с помощью Python (CVXOPT)

...

```
solution = solvers.lp(c, G.T, h, solver='glpk')
```

```
print('Status:', solution['status'])
```

```
print('Objective:', solution['primal objective'])
```

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

```
Status: optimal
Objective: -61000.0
x =
  [ 3.00e+01]
  [ 1.00e+01]
  [ 0.00e+00]
```

Теневые цены (shadow price)

- Теневая цена λ_i ограничения y_i (с соответствующим значением вектора правых частей b_i) показывает, *на сколько* изменится значение целевой функции в точке оптимума, если значение b_i будет увеличено на единицу (при условии, что не изменится состав базисных переменных).
- Допустим, есть ограничение, заключающееся в 40 часовой рабочей неделе. Теневая цена этого ограничения в точке оптимума покажет:
 - *На сколько* изменится значение целевой функции, если увеличить правую часть на 1.
 - Другими словами, сколько максимально можно доплачивать рабочим за дополнительный час, чтобы не ухудшить «прибыль».
- Направление изменения зависит от вида ограничения и решаемой задачи оптимизации:
 - Ослабление ограничения всегда делает значение ЦФ «лучше», усиление – наоборот
 - Для максимизации «лучше» означает больше, для минимизации – меньше и наоборот

Получение теневых цен в Python (CVXOPT)

```
...  
print(solution['z'])
```

Теневые цены

[1.20e+03]
[-0.00e+00]
[1.62e+02]
[-0.00e+00]
[-0.00e+00]
[-0.00e+00]
[-0.00e+00]
[-0.00e+00]
[-0.00e+00]
[1.25e+02]

Дефицитный
ресурс, поскольку
теневая цена не 0.

Если есть ограничения-равенства,
то их теневые цены находятся в
`solution['y']`

Исследование интервала осуществимости (1/2)

```
dh = matrix([1, 0, 0, 0, 0, 0, 0, 0, 0, 0]);      # приращение к вектору правых частей
```

```
solution1 = solvers.lp(c, G.T, h + dh, solver='glpk')
```

```
print('Status:', solution1['status'])
```

```
print('Objective:', -solution1['primal objective'],
```

```
      'delta:', -solution1['primal objective']-(-solution['primal objective']))
```

```
print(solution1['x'])
```

Status: optimal

Objective: 62200.0 delta: 1200.0

[3.10e+01]

[1.00e+01]

[0.00e+00]

Исследование интервала осуществимости (2/2)

```
prev_z = -solution['primal objective']
a = 1
while (True):
    solution_i = solvers.lp(c, G.T, h + dh*a, solver='glpk')
    if solution_i['status'] != 'optimal':
        print('Couldn't find a solution')
        break
    new_z = -solution_i['primal objective']
    delta_z = new_z - prev_z
    print('Increment %d: obj=%.2f delta=%.2f' % (a, new_z, delta_z))
    if abs(delta_z - 1200) > 1e-6:
        print('Basis changed at increment %d' % (a,))
        break
    prev_z = new_z
    a = a + 1
```

```
Increment 1: obj=62200.00 delta=1200.00
Increment 2: obj=63400.00 delta=1200.00
Increment 3: obj=64600.00 delta=1200.00
Increment 4: obj=65800.00 delta=1200.00
Increment 5: obj=67000.00 delta=1200.00
Increment 6: obj=68033.33 delta=1033.33
Basis changed at increment 6
```

Приведенная цена (reduced cost)

- Приведенная цена (*reduced cost*), в отличие от теневой цены, ассоциируется уже не с ограничениями, а с переменными.
- Приведенной ценой u_i небазисной переменной x_i является величина, на которую изменится значение целевой функции в точке оптимума, если x_i будет увеличено с 0 до 1 (войдет в базис), при условии, что это изменение мало.

Приведенная цена (reduced cost)

- 1) Для базисных переменных приведенная цена всегда 0.
- 2) Приведенную цену также можно назвать ценой возможности (opportunity cost). Предположим, нас вынудили произвести единицу x_i , продукта, который мы не собирались производить. Эта вынужденная возможность будет нам чего-то стоить, поскольку связана с добавлением нового ограничения. Насколько именно снизится значение ЦФ – и есть приведенная цена.
- 3) Приведенная цена u_i – это величина, на которую должен измениться коэффициент c_i перед x_i , чтобы x_i стал ненулевым в точке оптимума. Предположим, мы производим товары x_1, \dots, x_n , которые приносят нам доход c_1, \dots, c_n соответственно. Мы связаны определенными ограничениями, которые здесь не играют роли. Сформировав и решив задачу ЛП, чтобы оптимизировать прибыль, получаем оптимальный план производства: x_1^*, \dots, x_n^* , соответствующий прибыли z^* . Предположим, в этом плане $x_2^* = 0$. Очевидно, прибыль от производства товаров этого типа недостаточно велика, чтобы нам было выгодно производить их (а не какие-то другие товары). Тогда мы можем спросить себя: а какой должна быть прибыль от x_2 , чтобы нам все-таки было выгодно их производить, хотя бы в небольших количествах? Ответ $c_2 - u_2$ (или $c_2 + u_2$, в зависимости от того, как интерпретируется знак). Это означает, что прибыль должна увеличиться по крайней мере на размер reduced cost, чтобы стало выгодно производить этот продукт.

Приведенная цена (reduced cost)

Приведенная цена равна теневой цене для ограничений неотрицательности переменных.

Получение приведенных цен в Python (CVXOPT)

...

```
print(solution['z'])
```

```
[ 1.20e+03]  
[-0.00e+00]  
[ 1.62e+02]  
[-0.00e+00]  
[-0.00e+00]  
[-0.00e+00]  
[-0.00e+00]  
[-0.00e+00]  
[-0.00e+00]  
[ 1.25e+02]
```

Приведенные цены



Видим, что для единственной небазисной переменной, соответствующей продукту ПЗ, приведенная цена равна 125 руб. (для базисных она всегда равна нулю). То есть, если мы включим в план производства одну штуку продукта ПЗ, то выручка должна уменьшиться на 125 руб. И наоборот, если мы увеличим цену на продукт ПЗ хотя бы на 125 руб., то производить этот продукт станет выгодно.

Получение приведенной цены в GNU Octave

```
solution2 = solvers.lp(c - matrix([0, 0, 126]), G.T, h, solver='glpk')

print('Status:', solution2['status'])
print('Objective:', -solution2['primal objective'],
      'delta:', -solution2['primal objective']+solution['primal objective'])
print(solution2['x'])
```

Status: optimal

Objective: 61020.0 delta: 20.0

[1.50e+01]

[5.00e+00]

[2.00e+01]

Анализ чувствительности с помощью GNU Octave

Таблица

	x1 [C]	x2 [C]	x3 [C]	Неравенство	b
c	1200	2500	1400	-	max [-1]
y1	1	1	1	<= [U]	40
y2	0	0	1	<= [U]	20
y3	0	8	2	<= [U]	80
y4	1	0	0	<= [U]	35
y5	0	1	0	<= [U]	25
y6	0	0	1	<= [U]	30
y7	0,8	0,6	0,7	<= [U]	50

F – free,
U – upper,
S – equality
L - lower

Решение в GNU Octave

```
c = [1200 2500 1400]';
```

```
A = [1    1    1;  
      0    0    1;  
      0    8    2;  
      1    0    0;  
      0    1    0;  
      0    0    1;  
      0.8 0.6 0.7];
```

```
b = [40 20 80 35 25 30 50]';
```

```
[x_max, z_max, errnum] = glpk(c, A, b, zeros(3, 1), [], ...  
                              "UUUUUUUU", "CCC", -1)
```

```
x_max =  
  
      30  
      10  
       0  
  
z_max = 61000  
errnum = 0
```

Получение теневых цен в Octave (1/2)

```
[x_max, z_max, s, extra] = glpk(c, A, b, zeros(3, 1), [], ...  
                                "UUUUUUUU", "CCC", -1);
```


extra

extra =

scalar structure containing the fields:

lambda =

1.2000e+003
0.0000e+000
1.6250e+002
0.0000e+000
0.0000e+000
0.0000e+000
0.0000e+000



Дефицитный
ресурс, поскольку
теневая цена не 0.

redcosts =

0
0
-125

time = 0
mem = 0

Исследование интервала осуществимости (1/2)

```
db = [1 0 0 0 0 0 0 0]'; % приращение к вектору правых частей
[x_max, z_max] = glpk(c, A, b + db, zeros(3, 1), [], "UUUUUUU", "CCC", -1);
x_max =

    31
    10
     0

z_max = 62200
```

Исследование интервала осуществимости (2/2)

```
db = [1 0 0 0 0 0 0]';
prev_z = z_max;
a = 1;
while (1)
    [x_max, z_max, en] = glpk(c, A, b + a*db, zeros(3, 1), [], ...
        "UUUUUUUU", "CCC", -1);

    if en != 0
        printf("glpk() was not able to find a solution!\n");
        break;
    endif
    printf("Increment %d: z_max = %f delta = %f\n", a, z_max, z_max - prev_z);
    if abs(z_max - prev_z - 1200) > 1e-6
        printf("Basis changed at increment %d\n", a);
        break;
    endif
    prev_z = z_max;
    a = a + 1;
endwhile
```

```
Increment 1: z_max = 62200.000000 delta = 1200.000000
Increment 2: z_max = 63400.000000 delta = 1200.000000
Increment 3: z_max = 64600.000000 delta = 1200.000000
Increment 4: z_max = 65800.000000 delta = 1200.000000
Increment 5: z_max = 67000.000000 delta = 1200.000000
Increment 6: z_max = 68033.333333 delta = 1033.333333
Basis changed at increment 6
```

Получение приведенной цены в GNU Octave

```
[x_max, z_max, e, extra] = glpk(c, A, b, zeros(3, 1), ...  
                                [], "UUUUUUUU", "CCC", -1);
```

```
extra.redcosts
```

```
ans =
```

```
0
```

```
0
```

```
-125
```

Видим, что для единственной небазисной переменной, соответствующей продукту ПЗ, приведенная цена равна 125 руб. (для базисных она всегда равна нулю). То есть, если мы включим в план производства одну штуку продукта ПЗ, то выручка должна уменьшиться на 125 руб. И наоборот, если мы увеличим цену на продукт ПЗ хотя бы на 125 руб., то производить этот продукт станет выгодно.

Получение приведенной цены в GNU Octave

```
[x_max, z_max, e] = glpk(c + [0 0 126]', A, b, ...  
                        zeros(3,1), [], "UUUUUUUU", "CCC", -1)
```

```
x_max =
```

```
15
```

```
5
```

```
20
```

```
z_max = 61020
```

```
e = 0
```

Анализ чувствительности с помощью GMPL

GMPL. Версия 1

```
param p{j in 1..3};
param d{i in 1..3};
param c{i in 1..3};

var x{i in 1..3} >= 0;

maximize z : sum{i in 1..3} c[i]*x[i];

s.t. Prod1: x[1] + x[2] + x[3] <=
p[1];
s.t. Prod2: x[3] <= p[2];
s.t. Prod3: 8*x[2] + 2*x[3] <= p[3];
s.t. Demand{i in 1..3}: x[i] <= d[i];
s.t. Mat: 0.8*x[1] + 0.6*x[2] +
0.7*x[3] <= 50;
```

data;

```
param p:= 1 40 2 20 3 80;
param d:= 1 35 2 25 3 30;
param c:= 1 1200 2 2500 3 1400;
```

end;

Вызов:

```
glpsol -m factory_plan.mod -o solution.txt
```

GMPL. Версия 2

factory_plan.mod

```
# Виды продукции
set P := 1..3;
# Цеха
set W := 1..3;
# Технологические операции
set Ops dimen 2, within P cross W
    := {(1, 1),
        (2, 1), (2, 3),
        (3, 1), (3, 2), (3, 3)};
# Категории цехов
set W_Mass within W := {1, 2};
set W_Manual within W := W diff W_Mass;

# Виды продукции, обрабатываемые в цехе
set W_P{w in W} :=
    setof {(i, w) in Ops} i;
```

```
# Цена по видам продукции, руб.
param price{p in P};

# Спрос в неделю по видам продукции, шт.
param demand{p in P};

# Расход материала на производство продукции, кг
param cost{p in P};

# Производительность цехов, шт. в неделю
param prod{w in W_Mass};

# Затраты времени на единицу продукции, часы
param prod_m_cost{w in W_Manual, p in P};

# Фонд рабочего времени, часы
param prod_m{w in W_Manual};

# Поставки материалов в неделю, кг
param materials;
```

GMPL. Версия 2

...продолжение

Количество продукта, которое нужно произвести в неделю

```
var x{p in P} >= 0;
```

Ограничение для «поточных» цехов

```
s.t. mass{w in W_Mass}: sum{p in W_P[w]} x[p] <= prod[w];
```

Ограничение для цехов «ручной обработки»

```
s.t. manual{w in W_Manual}: sum{p in W_P[w]} x[p] * prod_m_cost[w, p] <= prod_m[w];
```

```
s.t. dem{p in P}: x[p] <= demand[p]; # Ограничение по спросу
```

```
s.t. mat: sum{p in P} x[p] <= materials; # Ограничение по материалам
```

```
maximize profit : sum{p in P} price[p] * x[p];
```

```
end;
```


GMPL. Версия 2

factory_plan.dat

```
data;
# Цена по видам продукции, руб.
param price := 1 1200 2 2500 3 1400;
# Спрос в неделю по видам продукции, шт.
param demand := 1 35 2 25 3 30;
# Расход материала на производство продукции, кг
param cost := 1 0.8 2 0.6 3 0.7;
# Производительность цехов, шт. в неделю
param prod := 1 40 2 20;
# Затраты времени на единицу продукции, часы
param prod_m_cost default 0 := [3, 2] 8 [3, 3] 2;
# Фонд рабочего времени, часы
param prod_m := 3 80;
# Поставки материалов в неделю, кг
param materials := 50;
end;
```

Вызов:

```
glpsol -m factory_plan.mod
       -d factory_plan.dat
       -o solution.txt
```

GMPL. Результат

No.	Row name	St	Activity	Lower bound	Upper bound	Marginal
1	mass[1]	NU	40		40	1200
2	mass[2]	B	0		20	
3	manual[3]	NU	80		80	162.5
4	dem[1]	B	30		35	
5	dem[2]	B	10		25	
6	dem[3]	B	0		30	
7	mat	B	40		50	
8	profit	B	61000			

No.	Column name	St	Activity	Lower bound	Upper bound	Marginal
1	x[1]	B	30	0		
2	x[2]	B	10	0		
3	x[3]	NL	0	0		-125

GMPL. Диапазон осуществимости

```
glpsol -m factory_plan.mod -d factory_plan.dat --ranges ranges.txt -o solution.txt
```

No.	Row name	St	Activity	Slack Marginal	Lower bound Upper bound	Activity range	Obj coef range	Obj value at break point	Limiting variable
1	mass[1]	NU	40.00000	.	-Inf	10.00000	-1200.00000	25000.00000	x[1]
				1200.00000	40.00000	45.00000	+Inf	67000.00000	dem[1]

...

GMPL. «Расширенный» анализ чувствительности

- Добавить в текст модели вывод значений объектов
`printf “%f,%f”, demand[3], profit >> “tmpf.out”;`
- Разбить файл параметров на два:
 - Постоянные параметры `factory_plan_const.dat`
 - Варьируемые параметры `factory_plan_var.dat`
- Запускать из скрипта, указывая оба файла параметров:
`glpsol -m factory_plan.mod`
`-d factory_plan_const.dat`
`-d factory_plan_var.dat`
- Анализировать содержимое файла `tmpf.out`