Элементы анализа чувствительности с использованием GNU Octave

Пономарев А.В.

Анализ чувствительности позволяет получить ответ на вопрос как изменится найденное оптимальное решение задачи ЛП, при *незначительном* изменении исходных данных. Как правило, интересует 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

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. Причем, если 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.

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

Версия 1.1 (19.03.2015)

Формальная постановка

Пусть x_i – количество единиц продукции i-того вида, которое необходимо произвести за неделю ($i \in \{1,..,3\}$).

Тогда условие задачи можно формально записать следующим образом:

$$1200x_1 + 2500x_2 + 1400x_3 \rightarrow max$$
$$x_1 + x_2 + x_3 \leq 40 \quad (1)$$
$$x_3 \leq 20 \quad (2)$$
$$8x_2 + 2x_3 \leq 80 \quad (3)$$
$$x_1 \leq 35 \quad (4)$$
$$x_2 \leq 25 \quad (5)$$
$$x_3 \leq 30 \quad (6)$$
$$0{,}8x_1 + 0{,}6x_2 + 0{,}7x_3 \leq 50 \quad (7)$$
$$x_{1,2,3} \geq 0$$

Ограничения (1)-(3) диктуются производительностью цехов, ограничения (4)-(6) обусловлены спросом на продукцию, а (7) выражает ограничение на использование материала.

Преобразуем постановку задачи в таблицу, из которой будет удобно заполнить аргументы для функции `glpk` (Таблица 4).

Таблица 4 – Подготовка исходных данных для вызова `glpk`

	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

Решим задачу с использованием функции `glpk` 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, status] = glpk(c, A, b, zeros(3, 1), [], "UUUUUUU", "CCC", -1)
x_max =

   30
   10
    0

z_max = 61000
status = 180
```

Код возврата (status) 180 говорит о том, что оптимальный план производства был найден. В x_max содержатся значения переменных x_1, x_2, x_3, а в z_max – максимально возможное значение целевой функции. Таким образом, оптимальный план предполагает выпуск 30 штук продукта П1 и 10 штук продукта П2, что даст выручку в размере 61 тыс. руб.

Замечание для пользователей GNU Octave 3.8.x. В одной из версий между 3.6.4 и 3.8.2 (к сожалению, более точной информации у меня нет) произошло изменение интерфейса функции glpk. Главным образом, это изменение коснулось способа сигнализации функцией об ошибках в ходе работы. Третье возвращаемое значение в 3.8.2 правильней называть не status, а код ошибки (в документации он обозначен как errnum). Нулевое значение кода ошибки говорит о том, что glpk отработал успешно (однако найденное решение не обязательно оптимально). Дополнительную информацию можно получить, проанализировав поле status четвертого возвращаемого значения – если glpk удалось найти оптимальное решение, это поле должно принять значение 5. Пример работы glpk в 3.8.2 для матриц, определенных выше:

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

   30
   10
    0

z_max =  61000
errnum = 0
```

Теневые цены

Теневая цена λ_i ограничения y_i (с соответствующим значением вектора правых частей b_i) показывает, на сколько вырастет значение целевой функции в точке оптимума, если ограничение b_i будет ослаблено 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 вырастет значение целевой функции, если ослабить ограничение на 1. Другими словами, сколько максимально можно доплачивать рабочим за дополнительный час, чтобы не ухудшить «прибыль».

Сведения для анализа чувствительности содержатся в структуре extra, возвращаемой функцией glpk(). Получим значение этой структуры для рассматриваемого примера (значения входных матриц оставляем без изменения):

```
[x_max, z_max, s, extra] = glpk(c, A, b, zeros(3, 1), [], "UUUUUUU", "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

    redcosts =
```

```
      0
      0
    -125

  time = 0
  mem = 0
```

Замечание для пользователей GNU Octave 3.8.x. Возвращаемые значения в 3.8.2 будут выглядеть немного иначе. В частности, в структуре `extra` появится еще одно поле `status`, значение которого позволяет различать ситуации нахождение оптимального решения (`status` = 5), отсутствия допустимых решений (`status` = 4), неограниченное возрастание или убывание целевой функции (`status` = 6) и еще ряд особых ситуаций. Приведенный выше пример в GNU Octave 3.8.2 трансформируется в:

```
[x_max, z_max, ec, extra] = glpk(c, A, b, zeros(3, 1), [], "UUUUUUU", "CCC", -1)
extra =

  scalar structure containing the fields:

    lambda =

       1.2000e+03
       0.0000e+00
       1.6250e+02
       0.0000e+00
       0.0000e+00
       0.0000e+00
       0.0000e+00

    redcosts =

         0
         0
      -125

    time = 0
    status =  5
```

Теневые цены содержатся в поле `lambda` этой структуры (`extra.lambda`). Элемент вектора `extra.lambda` с индексом *i* соответствует теневой цене *i*-того ограничения (заданного *i*-той строкой матрицы `A` при вызове `glpk`). В данном случае, мы видим, что у всех ограничений, кроме первого и третьего, теневые стоимости равны нулю, следовательно, ресурсы, соответствующие этим ограничениям, не являются дефицитными и увеличивать их количество смысла нет. Рассмотрим ограничения с ненулевой теневой ценой. Поскольку первое ограничение соответствует производительности первого цеха, то ненулевая теневая цена (1200 руб. – рублей, потому что именно в рублях измеряется наша целевая функция) для него означает, что при увеличении недельной производительности первого цеха (а именно для него составлено это ограничение) с 40 до 41 единицы продукции недельная выручка может увеличиться на 1200 руб. Это же число можно интерпретировать и иначе – если мы можем увеличить производительность цеха Ц1 на единицу, затратив меньше, чем 1200 руб., то это следует сделать, поскольку приведет к увеличению дохода. Ненулевая теневая цена третьего ограничения (162,50 руб.) означает, что при увеличении общего фонда рабочего времени сотрудников цеха Ц3 на 1 час (третье ограничение составлено для цеха Ц3 и его части измеряются в часах), выручка может вырасти на 162 руб. 50 коп. Опять же, альтернативно это означает, что если час рабочего времени сотрудников цеха Ц3 стоит меньше 162 руб. 50 коп., то можно доход можно увеличить.

Следует иметь в виду, что описанные оценки верны только для случая сохранения структуры решения. Если структура изменится, то есть, активизируются новые ограничения, то реальный прирост дохода от ослабления заданного ограничения может оказаться меньше. Попробуем решить задачу заново с измененными ограничениями. Начнем с первого:

```
db = [1 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
```

Действительно, значение выручки увеличилось на 1200 руб., как и «предсказывалось» теневой ценой первого ограничения. Попробуем перебирать различные приращения, чтобы найти то, которое приведет к изменению структуры решения (определяемому косвенно как приращение выручки менее, чем на теневую цену, то есть, менее, чем на 1200 руб.).

```
db = [1 0 0 0 0 0 0]';
prev_z = z_max;
a = 1;
while (1)
  [x_max, z_max, status] = glpk(c, A, b + a*db, zeros(3, 1), [], ...
                                "UUUUUUU", "CCC", -1);
  if status != 180
    printf("Not a single optimum. Special investigation needed.\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
```

Замечание для пользователей GNU Octave 3.8.x. Проверка того, что действительно было получено оптимальное решение трансформируется в:

```
  [x_max, z_max, ec, extra] = glpk(c, A, b + a*db, zeros(3, 1), [], ...
                                   "UUUUUUU", "CCC", -1);
  if ec != 0 || extra.status != 5
    printf("Not a single optimum. Special investigation needed.\n");
    break;
  endif
```

Результат выполнения скрипта:

```
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
```

Как видно из вывода, сгенерированного скриптом, без изменения состава базисных переменных данное ограничение может быть ослаблено на 5 единиц, что даст, в общей сложности увеличение выручки на 6 тыс. руб. Видно, также, что при дальнейшем увеличении выручка так же продолжает расти (просто теневые цены, вычисленные при исходных ограничениях, оказываются уже неактуальными). В принципе, подобный перебор можно продолжать и дальше, вплоть до момента, когда ослабление ограничения вообще перестанет влиять на значение целевой функции. В каком-то смысле, такой перебор является упрощенной версией анализа чувствительности, методом «грубой силы».

Исследование теневой цены третьего ограничения оставляю читателю в качестве упражнения.

Приведенная цена

Приведенная цена (*reduced cost*), в отличие от теневой цены, ассоциируется уже с ограничениями, а с переменными.
Приведенной ценой u_i небазисной переменной x_i является величина, на которую уменьшится значение целевой функции в точке оптимума, если x_i будет увеличено с 0 до 1 (войдет в базис), при условии, что это изменение мало.
Комментарии:
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, …, x_n, которые приносят нам доход c_1, …, 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^*, …, 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.
Значение приведенной цены находится в поле `redcosts` структуры `extra`, возвращаемой `glpk`:

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

ans =

     0
     0
  -125
```

Видим, что для единственной небазисной переменной, соответствующей продукту П3, приведенная цена равна 125 руб. (для базисных она всегда равна нулю). То есть, если мы включим в план производства одну штуку продукта П3, то выручка должна уменьшиться на 125 руб. И наоборот, если мы увеличим цену на продукт П3 хотя бы на 125 руб., то производить этот продукт станет выгодно. Для этого добавим нижнюю границу для переменной x_3 (параметр `lb` при вызове `glpk`):

```
[x_max, z_max, s] = glpk(c, A, b, [0 0 1]', [], "UUUUUUU", "CCC", -1)

x_max =
```

```
   29.2500
    9.7500
    1.0000

z_max =  60875
s =  180
```

Замечание для пользователей GNU Octave 3.8.x. Код ошибки (s) будет нулевой.

Действительно, выручка уменьшилась на 125 руб. Теперь попробуем увеличить цену на 125 руб., чтобы проверить, действительно ли при этих условиях станет выгодно производить продукт П3:

```
[x_max, z_max, s, extra] = glpk(c + [0 0 125]', A, b, zeros(3,1), ...
                                [], "UUUUUUU", "CCC", -1)
x_max =

   30
   10
    0

z_max =  61000
s =  180
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

    redcosts =

       0
       0
       0

    time = 0
    mem = 0
```

Замечание для пользователей GNU Octave 3.8.x. Код ошибки (s) будет нулевой, переменная `extra` будет содержать поле `status`, `extra.status` = 5.

Сам план не изменился, но обратите внимание, что приведенные цены теперь для всех продуктов равны нулю. Попробуем еще немного увеличить цену на третий продукт (на 126 руб.):

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

   15
    5
   20

z_max =  61020
s =  180
```

Замечание для пользователей GNU Octave 3.8.x. Код ошибки (s) будет нулевой.

В результате видим серьезное изменение плана производства (но незначительное приращение целевой функции) – теперь продукту П3 отдается предпочтение.