Прямой симплекс-метод (обоснование и реализация)

А. О. Махорин*

Август 2015 г.

Аннотация

В данной пояснительной записке рассмотрено математическое обоснование прямого (primal) модифицированного симплекс-метода, предназначенного для решения задач линейного программирования, а также приведено описание программных модулей, входящих в состав пакета GLPK, которые реализуют указанный метод на языке программирования ANSI C 89.

*Кафедра прикладной информатики, Московский авиационный институт. E-mail: <mao@gnu.org>.

Содержание

1 Общие сведения

1.1 Рабочий формат ЛП-задачи

В рассматриваемой реализации используется так называемый *рабочий формат* задачи линейного программирования (ЛП-задачи):

$$z = c^T x + c_0 \to \min \tag{1.1}$$

$$Ax = b \tag{1.2}$$

$$l \le x \le u \tag{1.3}$$

где $x = (x_k)$ — вектор переменных, z — целевая функция, $c = (c_k)$ — вектор коэффициентов целевой функции, c_0 — постоянный член целевой функции, $A = (a_{ik})$ — матрица коэффициентов ограничений, $b = (b_i)$ — вектор правых частей ограничений, $l = (l_k)$ — вектор нижних границ переменных, $u = (u_k)$ — вектор верхних границ переменных.

Если нижняя (верхняя) граница переменной x_k отсутствует, то формально считается, что $l_k = -\infty$ ($u_k = +\infty$). Переменная, у которой отсутствуют обе границы, называется *свободной (неограниченной) переменной*. Если $l_k = u_k$, то переменная x_k считается *фиксированной*.

В дальнейшем предполагается, что число переменных равно n, а число ограничений-равенств (1.2) равно m, поэтому матрица A имеет m строк и n столбцов. Также предполагается, что эта матрица имеет полный строчный ранг $\operatorname{rank}(A) = m$, т. е. что все строки A линейно независимы, откуда, в частности, следует, что $m \le n$.

1.2 Базисные решения

При решении ЛП-задачи симплекс-методом среди всевозможных допустимых решений, т. е. точек, удовлетворяющих всем ограничениям, особую роль играют так называемые *базисные решения*. Геометрически допустимые базисные решения соответствуют вершинам (крайним точкам) выпуклого многогранного множества[1] допустимых точек ЛП-задачи, где каждая вершина определяется как точка пересечения граней (гиперплоскостей), соответствующих активным[2] линейно независимым ограничениям.

Так как для ЛП-задачи (1.1)—(1.3) предполагается, что все ограничения (1.2) линейно независимы (матрица A имеет полный строчный ранг), то, являясь равенствами, все эти ограничения будут активны в любом базисном решении. Следовательно, для однозначного определения базисного решения как точки в n-мерном пространстве переменных $x = (x_k)$ к m ограничениям-равенствам (1.2) необходимо добавить еще $n - m$ активных ограничений из числа границ переменных (1.3).

Переменные, границы которых являются активными ограничениями в базисном решении, называются *небазисным (независимыми) переменными*. Остальные переменные с неактивными границами называются *базисными (зависимыми) переменными*. Поскольку нижняя и верхняя границы небазисной переменной не могут быть активны одновременно,[3] то в любом базисном решении всегда будет $n - m$ небазисных и m базисных переменных.

[1] Если это множество ограничено, то оно представляет собой выпуклый многогранник.

[2] В данном случае более правильным было бы называть эти ограничения *связывающими*.

[3] Случай фиксированной переменной, т. е. когда $l_k = u_k$, можно рассматривать как предельный случай двустороннего ограничения $l_k - \varepsilon \le x_k \le u_k + \varepsilon$ при $\varepsilon \to 0$.

Таким образом, базисное решение ЛП-задачи (1.1)—(1.3) однозначно определено, если указано, какие переменные являются базисными, а какие небазисными, и для каждой небазисной переменной, имеющей обе (нижнюю и верхнюю) границы, также указано какая именно из этих границ активна. При этом требуется, чтобы все n активных ограничений были линейно независимыми.

Перейдем к алгебраическому описанию базисных решений. Пусть

$$\Pi x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}, \ \Pi c = \begin{pmatrix} c_B \\ c_N \end{pmatrix}, \ \Pi l = \begin{pmatrix} l_B \\ l_N \end{pmatrix}, \ \Pi u = \begin{pmatrix} u_B \\ u_N \end{pmatrix}, \tag{1.4}$$

где Π — подходящая перестановочная матрица, $x_B = [(x_B)_i]$ — вектор базисных переменных, $x_N = [(x_N)_j]$ — вектор небазисных переменных, $c_B = [(c_B)_i]$ и $c_N = [(c_N)_j]$ — векторы коэффициентов целевой функции, $l_B = [(l_B)_i]$ и $l_N = [(l_N)_j]$ — векторы нижних границ, $u_B = [(u_B)_i]$ и $u_N = [(u_N)_j]$ — векторы верхних границ, соответственно, базисных и небазисных переменных. Соотношение (1.4) позволяет представить ограничения-равенства (1.2) в следующем эквивалентном виде:

$$A\Pi^T \Pi x = b \quad \Leftrightarrow \quad (B \,|\, N) \begin{pmatrix} x_B \\ x_N \end{pmatrix} = b \quad \Leftrightarrow \quad Bx_B + Nx_N = b, \tag{1.5}$$

где

$$(B \,|\, N) = A\Pi^T. \tag{1.6}$$

Здесь B — квадратная матрица порядка m, называемая *базисной матрицей* (или просто *базисом*). Эта матрица составлена из столбцов исходной матрицы коэффициентов ограничений A, соответствующих базисным переменным. Матрица N имеет m строк и $n - m$ столбцов, которыми являются столбцы A, соответствующие небазисным переменным.

Пусть $f = (f_j)$ — вектор активных границ небазисных переменных, где $f_j = (l_N)_j$ или $f_j = (u_N)_j$. Тогда базисное решение определяется следующей системой n уравнений, составленной из m активных ограничений-равенств (1.2) в форме (1.5) и $n - m$ активных границ небазисных переменных:

$$\begin{cases} Bx_B + Nx_N = b \\ x_N = f \end{cases} \tag{1.7}$$

Так как все активные ограничения предполагаются линейно независимыми, то матрица системы (1.7) невырожденная, откуда следует, что базисная матрица B также невырожденная. Это позволяет разрешить систему ограничений-равенств (1.2) в форме (1.5) относительно базисных переменных, чтобы явно выразить их через небазисные переменные:

$$x_B = -B^{-1}Nx_N + B^{-1}b \quad \Leftrightarrow \quad x_B = \Xi x_N + g, \tag{1.8}$$

где

$$\Xi = -B^{-1}N, \tag{1.9}$$

$$g = B^{-1}b. \tag{1.10}$$

Здесь $\Xi = (\xi_{ij})$ — $m \times (n-m)$-матрица, называемая *симплекс-таблицей*, $g = (g_i)$ — m-вектор, называемый *преобразованным вектором правых частей ограничений*.

Определения (1.9) и (1.10) с учетом соотношения (1.4) позволяют записать базисное решение как решение системы (1.7):

$$x = \Pi^T \begin{pmatrix} x_B \\ x_N \end{pmatrix} = \Pi^T \begin{pmatrix} \beta \\ f \end{pmatrix}, \tag{1.11}$$

где

$$\beta = (\beta_i) = -B^{-1}Nf + B^{-1}b = \Xi f + g \tag{1.12}$$

есть m-вектор значений базисных переменных в базисном решении.

Примечание. Может случиться, что небазисной переменной является свободная переменная[4] $(x_N)_j = x_k$, у которой отсутствуют обе нижняя и верхняя границы ($l_k = -\infty$ и $u_k = +\infty$). Тогда мы можем формально рассматривать такую переменную как сумму двух переменных $x_k = x'_k + x''_k$, где $x'_k \geq 0$ (неотрицательная переменная), $x''_k \leq 0$ (неположительная переменная). Переменные x'_k и x''_k не могут быть базисными одновременно, так как в противном случае в базисной матрице B были бы два одинаковых, а значит, линейно зависимых столбца. Поэтому, если x_k является базисной и ее текущее значение неотрицательно (неположительно), то это эквивалентно тому, что базисной является переменная x'_k (соответственно, x''_k), а переменная x''_k (соответственно, x'_k) является небазисной и поэтому находится на своей активной нулевой границе. Следовательно, если x_k является небазисной, то это эквивалентно тому, что обе переменные x'_k и x''_k небазисные, а значит, формально можно считать, что в этом случае значение переменной $(x_N)_j = x_k$ в базисном решении равно нулю, как если бы эта переменная имела активную нулевую границу.

Базисное решение является допустимым, если значения всех переменных удовлетворяют всем ограничениям ЛП-задачи. Так как соотношение (1.8) эквивалентно исходной системе ограничений-равенств (1.2), то любое базисное решение по определению удовлетворяет этим ограничениям. Кроме того, значения небазисных переменных в любом базисном решении совпадают с их границами, поэтому нарушение границ (1.3) возможно только для базисных переменных. Следовательно, базисное решение (1.11) является допустимым только если $l_B \leq \beta \leq u_B$, т. е. если значения всех базисных переменных находятся внутри их границ.

Всякое недопустимое базисное решение характеризуется тем, что хотя бы одна базисная переменная в этом решении нарушает свою нижнюю или верхнюю границу. Геометрически недопустимые базисные решения также представляют собой точки пересечения граней (гиперплоскостей), соответствующих активным линейно независимым ограничениям, однако эти точки лежат вне выпуклого многогранного множества допустимых решений ЛП-задачи.

1.3 Симплекс-метод

Симплекс-метод — это численный метод решения ЛП-задач, разработанный Дж. Данцигом (G. B. Dantzig) в 1947 г.[5]

С геометрической точки зрения общая идея симплекс-метода достаточно проста и заключается в следующем. Предположим, что мы находимся в некоторой вершине многогранного множества допустимых решений ЛП-задачи. Из этой вершины исходят *ребра*, ведущие в *смежные* вершины. Если движение вдоль какого-либо ребра приводит к улучшению целевой функции, то мы переходим в соответствующую смежную вершину с лучшим значением целевой функции, после чего повторяем все сначала до тех пор, пока это возможно. Поскольку число вершин многогранного множества с конечным числом граней конечно, то после выполнения конечного[6] числа шагов мы придем в вершину, из которой переход в смежную вершину с луч-

[4] Такую переменную более правильно было бы называть *супербазисной* по аналогии с супербазисными переменными в задачах с нелинейной целевой функцией.

[5] Впервые симплекс-метод был опубликован в работе [1]. См. также [2].

[6] При условии, что приняты необходимые меры, предотвращающие зацикливание, которое возможно, если имеются кратные (совпадающие) вершины.

шим значением целевой функции невозможен. Вершина, обладающая указанным свойством, будет оптимальным решением ЛП-задачи.

Рассмотрим алгебраическое описание симплекс-метода применительно к ЛП-задаче в формате (1.1)—(1.3).

Пусть у нас имеется допустимое базисное решение (1.11), соответствующее текущей вершине многогранного множества допустимых решений. Движение вдоль ребра по направлению к смежной вершине характеризуется тем, что какое-либо одно активное ограничение перестает быть активным, в то время как остальные активные ограничения остаются активными. Так как ограничения-равенства (1.2) активны в любом базисным решении, то указанным ограничением может быть только ограничение вида (1.3), т. е. активная граница какой-либо небазисной переменной $(x_N)_j$. Таким образом, движение вдоль ребра означает, что соответствующая этому ребру небазисная переменная начинает изменяться в допустимом направлении: увеличиваться, если активной была ее нижняя граница, или уменьшаться, если активной была ее верхняя граница.

Предположим, что небазисная переменная $(x_N)_j$ изменяется в допустимом направлении. В соответствии с (1.8) это приведет к изменению базисных переменных, а также целевой функции. Чтобы оценить суммарное изменение целевой функции (1.1), обусловленое изменением как небазисной, так и базисных переменных, запишем ее в следующем виде, используя соотношение (1.4):

$$z = c^T \Pi^T \Pi x + c_0 = \begin{pmatrix} c_B \\ c_N \end{pmatrix}^T \begin{pmatrix} x_B \\ x_N \end{pmatrix} + c_0 = c_B^T x_B + c_N^T x_N + c_0, \tag{1.13}$$

после чего подставим в (1.13) выражение для x_B из (1.8). В результате получим:

$$\begin{aligned} z &= c_B^T(-B^{-1}Nx_N + B^{-1}b) + c_N^T x_N + c_0 = \\ &= (c_N^T - c_B^T B^{-1} N)x_N + c_B^T B^{-1} b + c_0 = \\ &= (c_N - N^T B^{-T} c_B)^T x_N + (B^{-T} c_B)^T b + c_0 = \\ &= d^T x_N + d_0, \end{aligned} \tag{1.14}$$

где

$$d = c_N + \Xi^T c_B = c_N - N^T \pi, \tag{1.15}$$

$$\pi = B^{-T} c_B, \tag{1.16}$$

$$d_0 = \pi^T b + c_0. \tag{1.17}$$

Здесь $d = (d_j)$ — вектор так называемых *относительных оценок небазисных переменных*,[7] где относительная оценка d_j показывает суммарное (т. е. прямое и косвенное через базисные переменные) влияние небазисной переменной $(x_N)_j$ на целевую функцию при условии, что остальные небазисные переменные не изменяются. Вектор $\pi = (\pi_i)$ — это вектор так называемых *оценок правых частей ограничений* (или *симплексных множителей*),[8] где оценка правой части π_i показывает влияние изменения правой части i-го ограничения b_i на целевую функцию. Скалярная величина d_0 есть постоянный член преобразованной целевой функции (1.14).

[7] Компоненты вектора d суть значения множителей Лагранжа для активных границ небазисных переменных.
[8] Компоненты вектора π суть значения множителей Лагранжа для ограничений-равенств (1.2).

В случае минимизации движение вдоль выбранного ребра должно приводить к уменьшению целевой функции. Так как выбор ребра соответствует выбору небазисной переменной, это означает, что мы можем выбрать любую небазисную переменную с активной нижней границей, относительная оценка которой отрицательна, или любую небазисную переменную с активной верхней границей, относительная оценка которой положительна, поскольку изменение такой переменной в допустимом направлении даст уменьшение целевой функции. В случае свободной небазисной переменной достаточно, чтобы ее относительная оценка была ненулевой, так как свободная переменная может изменяться в любом направлении. Если при этом не найдется ни одной подходящей небазисной переменной, то текущее базисное решение соответствует минимуму целевой функции (является оптимальным). Заметим, что обычно имеется несколько подходящих небазисных переменных, поэтому возникает проблема выбора наиболее подходящей небазисной переменной.

Допустим, что выбрана подходящая небазисная переменная $(x_N)_q$ и эта переменная начинает изменяться в допустимом направлении (возрастать или убывать), при этом остальные небазисные переменные остаются на своих активных границах. Тогда в соответствии с (1.8) начнут изменяться базисные переменные x_B. Так как текущее базисное решение предполагается допустимым, то в начальный момент, когда $(x_N)_q$ находится на своей активной границе, все базисные переменные будут находиться внутри своих границ. Понятно, что если в процессе изменения $(x_N)_q$ некоторая базисная переменная $(x_B)_p$ первой достигнет своей (нижней или верхней) границы, либо сама небазисная переменная $(x_N)_q$ первой достигнет своей противоположной границы, то это будет означать, что мы достигли смежной вершины и дальнейшее изменение $(x_N)_q$ в том же направлении невозможно, так как это приведет к нарушению границы соответствующей переменной, а значит, к выходу за пределы допустимого множества.

Чтобы найти переменную, определяющую смежное базисное решение, положим

$$(x_N)_q = f_q + s\theta, \tag{1.18}$$

где f_q — активная граница небазисной переменной $(x_N)q$ в текущем базисном решении, $s = -\operatorname{sign} d_q$ ($s = +1$, если $d_q < 0$, т. е. $(x_N)_q$ возрастает, или $s = -1$, если $d_q > 0$, т. е. $(x_N)_q$ убывает), $\theta \geq 0$ — скалярный параметр луча, который исходит из текущей вершины и направление которого совпадает с направлением ребра, соответствующего переменной $(x_N)_q$. Заметим, что увеличение параметра θ начиная с нуля соответствует изменению $(x_N)_q$ в допустимом направлении начиная с ее активной границы f_q. Поскольку остальные небазисные переменные не изменяются, то с учетом (1.8) и (1.12) отдельная базисная переменная $(x_B)_i$ будет зависеть от θ следующим образом:

$$(x_B)_i = \beta_i + \xi_{iq} s\theta, \tag{1.19}$$

где β_i — значение $(x_B)_i$ в текущем базисном решении, ξ_{iq} — элемент симплекс-таблицы (1.9). Очевидно, что если $\xi_{iq}s < 0$, то $(x_B)_i$ убывает вдоль луча, а если $\xi_{iq}s > 0$, то $(x_B)_i$ возрастает вдоль луча. Поэтому $(x_B)_i$ достигнет своей нижней границы $(l_B)_i$ или верхней границы $(u_B)_i$ при $\theta = \theta_i \geq 0$, где

$$\theta_i = \begin{cases} \dfrac{(l_B)_i - \beta_i}{\xi_{iq}s}, & \text{если } \xi_{iq}s < 0 \\ \dfrac{(u_B)_i - \beta_i}{\xi_{iq}s}, & \text{если } \xi_{iq}s > 0 \end{cases} \tag{1.20}$$

а значит, *первой* достигнет своей границы та базисная переменная $(x_B)_p$, для которой

$$\theta_p = \min_i \theta_i. \tag{1.21}$$

Кроме того, небазисная переменная $(x_N)_q$ достигнет своей противоположной границы при $\theta = \theta_0 \geq 0$, где

$$\theta_0 = |(l_N)_q - (u_N)_q|. \tag{1.22}$$

Поэтому луч останется в допустимом множестве при $0 \leq \theta \leq \theta_{\min}$, где

$$\theta_{\min} = \min(\theta_p, \theta_0). \tag{1.23}$$

Таким образом, величина $\theta_{\min}$ позволяет идентифицировать переменную, которая первой достигает своей границы и тем самым определяет смежное базисное решение: если $\theta_{\min} = \theta_p$, то это базисная переменная $(x_B)_p$, а если $\theta_{\min} = \theta_0$, то это небазисная переменная $(x_N)_q$. Нужно также заметить, что возможен случай $\theta_{\min} = \infty$, который означает, что ЛП-задача имеет неограниченный оптимум (минимум), т. е. в этом случае возможно неограниченное уменьшение целевой функции вдоль соответствующего луча.

Если $\theta_{\min} = \theta_p$, то в смежном базисном решении та (нижняя или верхняя) граница переменной $(x_B)_p$, которая препятствует дальнейшему изменению $(x_N)_q$, т. е. которая определяет величину θ_p, становится активной, а текущая активная граница f_q переменной $(x_N)_q$, наоборот, перестает быть активной. Другими словами, в смежном базисе переменные $(x_B)_p$ и $(x_N)_q$ меняются ролями: $(x_B)_p$ становится небазисной (выходит из базиса), а $(x_N)_q$ становится базисной (входит в базис). Если же $\theta_{\min} = \theta_0$, то в смежном базисе переменная $(x_N)_q$ остается небазисной, однако активной становится ее противоположная граница. Зная величину $\theta_{\min}$ можно также определить величину приращения целевой функции при переходе от текущего базиса к смежному:

$$\Delta z = -|d_q \theta_{\min}| \leq 0. \tag{1.24}$$

Выбор базисной переменной может оказаться неоднозначным, если несколько переменных одновременно (т. е. при одном и том же значении $\theta = \theta_{\min}$) достигают своих границ, причем эта неоднозначность обычно усугубляется ситуацией $\theta_{\min} = 0$. В последнем случае при переходе к смежному базису целевая функция не изменяется, поэтому, если не принимать специальных мер, возможно зацикливание метода, когда происходит возврат к одному из предыдущих базисных решений.

В заключение рассмотрим принципиальную схему симплекс-метода применительно к ЛП-задаче (1.1)—(1.3), полагая, что задано некоторое начальное допустимое базисное решение (указано, какие переменные являются базисными, какие небазисными, и для небазисных переменных указаны их активные границы).

СИМПЛЕКС-МЕТОД

Шаг 1. Вычислить $\beta = (\beta_i)$ — вектор текущих значений базисных переменных $x_B = [(x_B)_i]$ (1.12), $d = (d_j)$ — вектор относительных оценок небазисных переменных $x_N = [(x_N)_j]$ (1.16).

Шаг 2. Выбрать подходящую небазисную переменную $(x_N)_q$, изменение которой в допустимом направлении уменьшает целевую функцию, т. е. относительная оценка d_q которой имеет «неправильный» знак. Небазисная переменная $(x_N)_j$ считается подходящей, если:

$d_j < 0$ и $(x_N)_j$ имеет активную нижнюю границу;

$d_j > 0$ и $(x_N)_j$ имеет активную верхнюю границу;

$d_j \neq 0$ и $(x_N)_j$ является свободной (неограниченной) переменной.

Фиксированные небазисные переменные не могут изменяться и поэтому не включаются в число подходящих.

Если выбор невозможен, то СТОП — текущее базисное решение является оптимальным.

Шаг 3. Определить $\theta_{\min}$ — абсолютную величину приращения выбранной небазисной переменной $(x_N)_q$ в смежном базисе (1.23).

Если $\theta_{\min} = \infty$, то СТОП — ЛП-задача имеет неограниченный минимум.

Шаг 4. Если $\theta_{\min} = \theta_p$ (1.21), то пометить переменную $(x_B)_p$ как небазисную с активной нижней или верхней границей в зависимости от выбора $\theta_i = \theta_p$ (1.20), а переменную $(x_N)_q$ пометить как базисную. Если же $\theta_{\min} = \theta_0$ (1.22), то изменить текущую активную границу переменной $(x_N)_q$ на противоположную.

Вернуться к шагу 1. ■

1.4 Табличный симплекс-метод

Табличный симплекс-метод — это вариант симплекс-метода, в котором для определения всех необходимых величин используется таблица (двумерный массив), соответствующая так называемой *преобразованной* (или *текущей*) *задаче*.

Преобразованная задача получается из исходной ЛП-задачи посредством эквивалентного линейного преобразования,[9] которое применяется для того, чтобы явно выразить целевую функцию и базисные переменные через небазисные переменные.

Так, для ЛП-задачи в формате (1.1)—(1.3) преобразованная задача имеет следующий формат:

$$z = d^T x_N + d_0 \to \min \tag{1.25}$$

$$x_B = \Xi x_N + g \tag{1.26}$$

$$l_B \le x_B \le u_B \tag{1.27}$$

$$l_N \le x_N \le u_N \tag{1.28}$$

где x_B — вектор базисных переменных, x_N — вектор небазисных переменных, d — вектор относительных оценок небазисных переменных (1.15), d_0 — постоянный член преобразованной целевой функции (1.17), Ξ — симплекс-таблица (1.9), g — преобразованный вектор правых частей ограничений (1.10).

Существенная особенность преобразованной задачи состоит в том, что ее основные компоненты (d, d_0, Ξ, g) относятся к текущему базису, поэтому всякий раз при переходе к смежному базису необходимо выполнять пересчет этих компонент.

Главным преимуществом табличного симплекс-метода является простота его реализации, поскольку вся необходимая информация доступна в явном виде.[10] Однако для практического решения ЛП-задач табличный симплекс-метод в настоящее время почти не применяется в основном из-за его низкой вычислительной эффективности, особенно при решении разреженных ЛП-задач высокой размерности. Например, на каждой итерации табличного симплекс-метода

[9] Это преобразование, которое можно выполнить, используя метод исключения Гаусса—Жордана, представляет собой умножение B^{-1} слева на исходную матрицу коэффициентов ограничений (1.6) $A\Pi^T = (B \,|\, N)$, в результате чего получается матрица $B^{-1}(B \,|\, N) = (I \,|\, B^{-1}N) = (I \,|\, -\Xi)$, соответствующая (1.26).

[10] Наиболее отчетливо это проявляется для ЛП-задач в стандартном формате, где вместо границ общего вида (1.3) используются условия неотрицательности $x \ge 0$: в этом случае все небазисные переменные имеют нулевые активные границы, поэтому текущее значение целевой функции равно d_0, а текущие значения базисных переменных равны элементам вектора g.

необходимо выполнять пересчет всей симплекс-таблицы Ξ, хотя для выбора базисной переменной нужен всего один столбец этой матрицы. При этом даже если исходная матрица A является достаточно разреженной, заполнение симплекс-таблицы $\Xi = -B^{-1}N$ в общем случае может быть гораздо выше.

В настоящее время наиболее широко используется вариант, называемый *модифицированным симплекс-методом* (см. следующий раздел). В этом варианте компоненты преобразованной задачи не хранятся в явном виде, а все используемые величины вычисляются только по мере необходимости, что позволяет существенно повысить вычислительную эффективность метода. Тем не менее, модифицированный симплекс-метод также использует компоненты преобразованной задачи, поэтому имеет смысл рассмотреть здесь формулы табличного симплекс-метода для пересчета компонент преобразованной задачи (1.25)—(1.28) при переходе к смежному базису, поскольку эти формулы можно использовать для вывода формул пересчета других величин, связанных с текущим базисным решением.

Чтобы упростить вывод указанных формул, заметим, что строку целевой функции (1.25) можно рассматривать как строку свободной базисной переменной $(x_B)_0 = z$. Аналогично, преобразованный вектор правых частей ограничений g можно рассматривать как столбец фиксированной небазисной переменной $(x_N)_0 = 1$. Это позволяет объединить основные компоненты (d, d_0, Ξ, g) в *расширенную симплекс-таблицу*:[11]

$$\Xi^* = (\xi_{ij}) = \begin{pmatrix} d_0 & d^T \\ g & \Xi \end{pmatrix}, \tag{1.29}$$

а также объединить равенства (1.25) и (1.26) в одно равенство:

$$x_B = \Xi^* x_N. \tag{1.30}$$

Допустим, что базисная переменная $(x_B)_p$ становится небазисной и занимает место небазисной переменной $(x_N)_q$, которая, в свою очередь, становится базисной и занимает место переменной $(x_B)_p$. Нас интересуют формулы пересчета элементов расширенной симплекс-таблицы Ξ^* при переходе к смежному базису.

Чтобы выразить новую базисную переменную $(\overline{x}_B)_p = (x_N)_q$ через новый набор небазисных переменных, включающий новую небазисную переменную $(\overline{x}_N)_q = (x_B)_p$, преобразуем сначала p-ю (ведущую) строку системы равенств (1.30):

$$(x_B)_p = \sum_{j=0}^{n-m} \xi_{pj}(x_N)_j = \xi_{pq}(x_N)_q + \sum_{j \neq q} \xi_{pj}(x_N)_j,$$

откуда

$$(x_N)_q = \frac{1}{\xi_{pq}}(x_B)_p - \sum_{j \neq q} \frac{\xi_{pj}}{\xi_{pq}}(x_N)_j. \tag{1.31}$$

Исключим теперь $(x_N)_q$ из остальных строк системы равенств (1.30), для чего подставим $(x_N)_q$ из (1.31) в каждую i-ю строку ($i \neq p$):

[11] Эта расширенная симплекс-таблица, представленная в виде двумерного массива, является той основной структурой данных, которая используется в табличном симплекс-методе.

$$
\begin{aligned}
(x_B)_i &= \sum_{j=0}^{n-m} \xi_{ij}(x_N)_j = \xi_{iq}(x_N)_q + \sum_{j\neq q} \xi_{ij}(x_N)_j = \\
&= \frac{\xi_{iq}}{\xi_{pq}}(x_B)_p - \sum_{j\neq q} \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}(x_N)_j + \sum_{j\neq q} \xi_{ij}(x_N)_j = \\
&= \frac{\xi_{iq}}{\xi_{pq}}(x_B)_p + \sum_{j\neq q} \left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)(x_N)_j.
\end{aligned} \tag{1.32}
$$

Коэффициенты при переменных в равенствах (1.31) и (1.32) являются элементами новой расширенной симплекс-таблицы $\overline{\Xi}^* = (\overline{\xi}_{ij})$ для смежного базиса, выраженные через элементы исходной расширенной симплекс-таблицы $\Xi^* = \xi_{ij}$ для текущего базиса. Таким образом, получаем следующие формулы пересчета:

$$
\overline{\xi}_{pq} = \frac{1}{\xi_{pq}}, \tag{1.33}
$$

$$
\overline{\xi}_{pj} = -\frac{\xi_{pj}}{\xi_{pq}} = -\overline{\xi}_{pq}\xi_{pj}, \quad j \neq q. \tag{1.34}
$$

$$
\overline{\xi}_{iq} = \frac{\xi_{iq}}{\xi_{pq}} = \overline{\xi}_{pq}\xi_{iq}, \quad i \neq p. \tag{1.35}
$$

$$
\overline{\xi}_{ij} = \xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}} = \xi_{ij} - \overline{\xi}_{iq}\xi_{pj} = \xi_{ij} + \xi_{iq}\overline{\xi}_{pj}, \quad i \neq p, j \neq q. \tag{1.36}
$$

Нулевая строка расширенной симплекс-таблицы (1.29) соответствует строке целевой функции в преобразованной задаче (1.25), т. е. $z = (x_B)_0$. Поэтому формулы пересчета относительных оценок $d_j = \xi_{0j}$ и постоянного члена $d_0 = \xi_{00}$ совпадают с формулами (1.35) и (1.36) при $i = 0$:

$$
\overline{d}_q = \frac{d_q}{\xi_{pq}} = \overline{\xi}_{pq}d_q, \tag{1.37}
$$

$$
\overline{d}_j = d_j - \frac{d_q\xi_{pj}}{\xi_{pq}} = d_j - \overline{d}_q\xi_{pj} = d_j + d_q\overline{\xi}_{pj}, \quad j \neq q. \tag{1.38}
$$

2 Модифицированный симплекс-метод

Модифицированный симплекс-метод [3, 4, 5] — это вариант симплекс-метода, в котором для вычисления всех необходимых величин используется подходящее представление текущей базисной матрицы B (1.6), позволяющее эффективно выполнять основные операции с этой матрицей (подразд. 2.1). В отличие от табличного симплекс-метода, (см. подразд. 1.4), преобразованная задача (1.25)—(1.28), за исключением ее отдельных компонент, в явном виде не хранится, что позволяет существенно повысить вычислительную эффективность симплекс-метода, особенно при решении разреженных ЛП-задач большой размерности.

2.1 Факторизация базисной матрицы

Представление текущей базисной матрицы B, используемое в модифицированном симплекс-методе, обычно (хотя и не обязательно) имеет вид матричной факторизации, где B представлена в виде произведения матриц специальной структуры.[12] Поэтому указанное представление называется в дальнейшем *факторизацией базисной матрицы.*

С точки зрения симплекс-метода вид конкретной факторизации базисной матрицы является несущественным. Однако предполагается, что используемая факторизация обеспечивает эффективное выполнение следующих четырех основных операций:

1. *Начальное вычисление факторизации.* Исходными данными этой операции является заданная матрица B. Результатом является вычисленная факторизация этой матрицы.

2. *Прямое преобразование (FTRAN).* Эта операция представляет собой решение системы $Bx = y$ (или, что то же самое, вычисление вектора $x = B^{-1}y$) для заданного вектора y.[13]

3. *Обратное преобразование (BTRAN).* Эта операция представляет собой решение системы $B^T x = y$ (или, что то же самое, вычисление вектора $x = B^{-T}y$) для заданного вектора y.

4. *Пересчет факторизации после замены столбца.* Исходными данными этой операции является факторизация текущей матрицы B и заданный вектор, который заменяет некоторый столбец в текущей матрице, что дает новую матрицу $\overline{B}$. Результатом операции является факторизация новой матрицы $\overline{B}$.

Примечание. Конкретная реализация факторизации базисной матрицы обычно ограничивает число пересчетов факторизации (например, не более 50 или 100), что обусловлено, в основном, техническими причинами (накопление вычислительных погрешностей, переполнение массивов, чрезмерное заполнение, и т. п.). Однако это ограничение не является критическим для модифицированного симплекс-метода, так как всегда имеется возможность выполнить начальное вычисление факторизации.

Реализация факторизации базисной матрицы практически не связана с реализацией симплекс-метода и представляет собой самостоятельную проблему, которая здесь не обсуждается.

В заключение рассмотрим формулы преобразования базисной матрицы при замене столбца. Эти формулы не зависят от вида конкретной факторизации и поэтому их можно использовать для пересчета других величин, связанных с базисной матрицей.

Итак, пусть p-й столбец матрицы B заменяется столбцом v, в результате чего получается матрица $\overline{B}$.

[12] Примером факторизации может служить треугольное разложение.

[13] Здесь x и y — это произвольные векторы, не связанные с ЛП-задачей.

Так как B — невырожденная матрица, то имеет место равенство

$$B^{-1}B = I. \tag{2.1}$$

Определим матрицу H следующим образом:

$$B^{-1}\overline{B} = H. \tag{2.2}$$

Поскольку $\overline{B}$ отличается от B только p-м столбцом, то из сопоставления (2.1) и (2.2) следует, что H отличается от единичной матрицы I также только p-м столбцом и поэтому имеет вид:

$$H = \begin{pmatrix} 1 & & h_{1p} & & \\ & \ddots & \vdots & & \\ & & 1 \quad h_{p-1,p} & & \\ & & h_{pp} & & \\ & & h_{p+1,p} \quad 1 & & \\ & & \vdots & \ddots & \\ & & h_{mp} & & 1 \end{pmatrix}, \tag{2.3}$$

где p-й столбец этой матрицы $H_p = (h_{ip})$ удовлетворяет равенству

$$B^{-1}v = H_p. \tag{2.4}$$

Умножая слева B на обе части равенства (2.2), получим основную формулу преобразования базисной матрицы:

$$\overline{B} = BH. \tag{2.5}$$

Формулы преобразования для обратных матриц непосредственно следуют из (2.5):

$$\overline{B}^{-1} = H^{-1}B^{-1}, \tag{2.6}$$

$$\overline{B}^{-T} = B^{-T}H^{-T}, \tag{2.7}$$

где

$$H^{-1} = \begin{pmatrix} 1 & & -h_{1p}/h_{pp} & & \\ & \ddots & \vdots & & \\ & & 1 \quad -h_{p-1,p}/h_{pp} & & \\ & & 1/h_{pp} & & \\ & & -h_{p+1,p}/h_{pp} \quad 1 & & \\ & & \vdots & \ddots & \\ & & -h_{mp}/h_{pp} & & 1 \end{pmatrix}. \tag{2.8}$$

(Таким образом, матрица $\overline{B}$ будет невырожденной только в том случае, если диагональный элемент h_{pp} матрицы H отличен от нуля.)

Примечание. Рассмотренные формулы можно использовать для достаточно простой реализации факторизации в виде так называемого *«эта-файла»* [4, 6]. В этой реализации текущая базисная матрица B хранится в следующем факторизованном виде:

$$B = B_0H_1 \dots H_k, \tag{2.9}$$

где B_0 — начальная базисная матрица (т. е. после выполнения операции вычисления факторизации $B = B_0$), представленная в подходящем факторизованном виде (например, в виде обычного треугольного разложения), позволяющем эффективно решать системы вида $B_0 x = y$ и $B_0^T x = y$; $H_1, \ldots, H_k$ — матрицы вида (2.3), отражающие замену столбцов в B_0 и хранимые в виде последовательного файла (массива), называемого «эта-файл» (так как буква H похожа на греческую «эта»). Из (2.9) следует, что:

$$x = B^{-1}y = H_k^{-1} \ldots H_1^{-1} B_0^{-1} y, \tag{2.10}$$

т. е. операцию FTRAN можно выполнить, решая вначале систему с матрицей B_0 и затем последовательно с матрицами $H_1, \ldots, H_k$ («эта-файл» просматривается в прямом порядке, откуда название операции — прямое преобразование). Транспонируя обе части формулы (2.10), получим:

$$x = B^{-T}y = B_0^{-T} H_1^{-T} \ldots H_k^{-T} y, \tag{2.11}$$

т. е. операцию BTRAN можно выполнить, решая последовательно системы с матрицами $H_k^T, \ldots, H_1^T$ и затем с матрицей B_0^T («эта-файл» просматривается в обратном порядке, откуда название операции — обратное преобразование). Наконец, операция замены столбца представляет собой вычисление и добавление очередной матрицы H_{k+1} к факторизации (2.9), т. е. в конец «эта-файла». (Отметим, что для факторизации B_0 также можно использовать «эта-файл», поскольку любую невырожденную матрицу можно получить, последовательно заменяя столбцы единичной матрицы.)

2.2 Основные операции

Анализ принципиальной схемы симплекс-метода (см. подразд. 1.3) позволяет выделить следующие основные операции, которые необходимы при решении ЛП-задачи (1.1)—(1.3) модифицированным симплекс-методом.

1. *Вычисление значений базисных переменных* $\beta = (\beta_i)$. Вектор β необходим для определения величины $\theta_{\min}$ (1.23), т. е. для выбора базисной переменной, которую следует исключить из числа базисных. Кроме того, этот вектор вместе с вектором активных границ (т. е. текущих значений) небазисных переменных f определяет текущее допустимое базисное решение (1.11). Из (1.12) следует, что вектор β можно вычислить, используя операцию FTRAN (см. подразд. 2.1), по формуле:

$$\beta = B^{-1}(b - Nf), \tag{2.12}$$

где b — вектор правых частей ограничений (1.2), N — матрица (1.6), составленная из столбцов матрицы коэффициентов ограничений A, соответствующих небазисным переменным x_N.

2. *Вычисление симплексных множителей* $\pi = (\pi_i)$. Вектор π необходим для вычисления относительных оценок небазисных переменных. Этот вектор можно вычислить, используя операцию BTRAN (см. подразд. 2.1), по формуле (1.16):

$$\pi = B^{-T} c_B, \tag{2.13}$$

где c_B — вектор коэффициентов целевой функции при базисных переменных x_B (1.4).

3. *Вычисление относительных оценок* $d = (d_j)$. Вектор d необходим для выбора небазисной переменной, которую следует включить в число базисных. Этот вектор можно вычислить по формуле (1.15):

$$d = c_N - N^T \pi, \tag{2.14}$$

где c_N — вектор коэффициентов целевой функции при небазисных переменных x_N (1.4). Заметим, что указанная формула позволяет эффективно вычислять относительные оценки отдельных небазисных переменных. Пусть $(x_N)_j = x_k$ — некоторая небазисная переменная. Тогда ее относительная оценка есть $d_j = e_j^T d$, где e_j — j-й столбец единичной матрицы. Поэтому:

$$d_j = e_j^T c_N - e_j^T N^T \pi = (c_N)_j - N_j^T \pi = c_k - A_k^T \pi, \tag{2.15}$$

где $(c_N)_j = c_k$ — коэффициент целевой функции и $N_j = A_k$ — столбец матрицы коэффициентов ограничений, соответствующие небазисной переменной $(x_N)_j = x_k$.

4. *Вычисление столбца симплекс-таблицы* $\Xi_q = (\xi_{iq})$, который соответствует выбранной небазисной переменной $(x_N)_q$. Вектор-столбец Ξ_q совместно с вектором текущих значений базисных переменных β необходим для определения величины $\theta_{\min}$ (см. выше). Так как $\Xi_q = \Xi e_q$, где e_q — q-й столбец единичной матрицы, то из (1.9) следует, что столбец Ξ_q можно вычислить, используя операцию FTRAN, по формуле:

$$\Xi_q = -B^{-1} N e_q = -B^{-1} N_q, \tag{2.16}$$

где N_q — столбец матрицы коэффициентов ограничений A, соответствующий небазисной переменной $(x_N)_q$.

5. *Переход к смежному базису.* Эта операция состоит в замене столбца B_p текущей базисной матрицы B столбцом N_q текущей матрицы N, где индексы p и q определяются выбранными базисной $(x_B)_p$ и небазисной $(x_N)_q$ переменными, в результате чего получается новая базисная матрица $\overline{B}$, соответствующая смежному базису. Поскольку базисная матрица не хранится явно, а используется ее факторизация, то выполнение данной операции сводится к пересчету факторизации базисной матрицы (см. подразд. 2.1).

Все перечисленные операции допускают эффективное выполнение, если матрица коэффициентов ограничений A доступна по столбцам.[14]

В заключение отметим, что если придерживаться принципиальной схемы симплекс-метода из подразд. 1.3, то каждая итерация модифицированного симплекс-метода требует выполнения двух операций FTRAN, одной операции BTRAN, и одного пересчета факторизации текущей базисной матрицы после замены столбца.

2.3 Пересчет компонент базисного решения

При решении ЛП-задач большой размерности число итераций симплекс-метода может достигать десятков и сотен тысяч. Поэтому для повышения вычислительной эффективности модифицированного симплекс-метода на каждой итерации вместо прямого вычисления основных компонент базисного решения по формулам (2.12), (2.13) и (2.14) выгоднее выполнять их пересчет, т. е. использовать известные значения этих компонент в текущем базисе для вычисления их значений в смежном базисе. Здесь необходимо отметить, что подобный пересчет неизбежно сопровождается накоплением ошибок округления, поэтому время от времени следует использовать прямое вычисление этих компонент (например, всякий раз после начального вычисления факторизации базисной матрицы).

[14] Различие между строчным и столбцовым представлениями матрицы является существенным только если для хранения матрицы используется разреженный формат.

Пересчет значений базисных переменных $\beta = (\beta_i)$. Пусть переменная $(x_B)_p$ выходит из базиса на свою активную границу $t = (l_B)_p$ или $t = (u_B)_p$, а переменная $(x_N)_q$ входит в базис. Так как остальные небазисные переменные не изменяются, то вектор приращений небазисных переменных при переходе к смежному базису есть

$$\Delta x_N = \Delta(x_N)_q e_q, \tag{2.17}$$

где $\Delta(x_N)_q$ — приращение переменной $(x_N)_q$, e_q — q-й столбец единичной матрицы. Из соотношения (1.8) следует, что в этом случае вектор приращений базисных переменных равен

$$\Delta x_B = \Xi \Delta x_N = \Delta(x_N)_q \Xi e_q = \Delta(x_N)_q \Xi_q, \tag{2.18}$$

где $\Xi_q = (\xi_{iq})$ — q-й столбец текущей симплекс-таблицы. В частности, для отдельной базисной переменной имеем:

$$\Delta(x_B)_i = \xi_{iq} \Delta(x_N)_q. \tag{2.19}$$

Зная приращение переменной $(x_B)_p$, которое, очевидно, равно

$$\Delta(x_B)_p = t - \beta_p, \tag{2.20}$$

где β_p — значение этой переменной в текущем базисе, из формулы (2.19) при $i = p$ следует, что соответствующее приращение переменной $(x_N)_q$ равно[15]

$$\Delta(x_N)_q = \frac{\Delta(x_B)_p}{\xi_{pq}} = \frac{t - \beta_p}{\xi_{pq}}. \tag{2.21}$$

А так как в смежном базисе $(x_N)_q$ занимает место $(x_B)_p$, то

$$\overline{\beta}_p = f_q + \Delta(x_N)_q, \tag{2.22}$$

где f_q — активная граница переменной $(x_N)_q$ в текущем базисе. Остальные базисные переменные остаются базисными в смежном базисе, поэтому из (2.19) следует, что

$$\overline{\beta}_i = \beta_i + \Delta(x_B)_i = \beta_i + \xi_{iq} \Delta(x_N)_q, \ i \neq p. \tag{2.23}$$

В специальном случае, когда в смежном базисе переменная $(x_N)_q$ переходит со своей текущей активной границы на противоположную, для определения приращения этой переменной вместо формулы (2.21) следует использовать формулу

$$\Delta(x_N)_q = s[(u_N)_q - (l_N)q], \tag{2.24}$$

где $s = +1$, если $(x_N)_q$ возрастает, или $s = -1$, если $(x_N)_q$ убывает. При этом значения *всех* базисных переменных пересчитываются по формуле (2.23).

Использование формул пересчета (2.22) и (2.23) дает возможность сэкономить одну операцию FTRAN на каждой итерации модифицированного симплекс-метода, поскольку используемый в этих формулах столбец симплекс-таблицы Ξ_q уже будет вычислен ранее (он требуется для определения величины $\theta_{\min}$).

[15] Понятно, что если выбор переменной $(x_B)_p$ выполнялся на основе величины $\theta_{\min}$ (см. принципиальную схему симплекс-метода, подразд. 1.3), то $\Delta(x_N)_q = s\theta_{\min}$.

Пересчет относительных оценок $d = (d_j)$. Формулы пересчета относительных оценок небазисных переменных — это формулы (1.37) и (1.38) (см. подразд 1.4):

$$\overline{d}_q = \frac{d_q}{\xi_{pq}}, \tag{2.25}$$

$$\overline{d}_j = d_j - \xi_{pj}\overline{d}_q,\ j \neq q. \tag{2.26}$$

Заметим, что в специальном случае, когда в смежном базисе небазисная переменная $(x_N)_q$ переходит со своей текущей активной границы на противоположную, *все* относительные оценки, включая оценку d_q, остаются *неизменными*.

Примечание. Так как пересчет сопровождается ошибками округления, то перед вычислением $\overline{d}_q$ по формуле (2.25) целесообразно использовать уже вычисленный столбец Ξ_q текущей симплекс-таблицы для прямого, а значит, более точного вычисления d_q в текущем базисе. Поскольку $d_q = e_q^T d$, где e_q — q-й столбец единичной матрицы, то необходимая формула получается умножением слева e_q^T на обе части равенства (1.15):

$$d_q = (c_N)_q + \Xi_q^T c_B. \tag{2.27}$$

Для пересчета оценок по указанным формулам требуется p-я строка симплекс-таблицы $\xi_p = (\xi_{pj})$ в текущем базисе. Так как $\xi_p = \Xi^T e_p$, где e_p — p-й столбец единичной матрицы, то из (1.9) следует, что вектор ξ_p можно вычислить по формуле:

$$\xi_p = -N^T B^{-T} e_p = -N^T \rho, \tag{2.28}$$

где вектор ρ есть p-я строка обратной матрицы B^{-1}, который можно вычислить, используя операцию BTRAN:

$$\rho = B^{-T} e_p. \tag{2.29}$$

Сравнивая формулы (2.28) и (2.29) с формулами прямого вычисления оценок (2.13) и (2.14) можно заметить, что они имеют идентичную структуру,[16] поэтому на первый взгляд пересчет относительных оценок не позволяет повысить эффективность вычислений по сравнению с их прямым вычислением. Однако, в отличие от вектора c_B в формуле (2.13), вектор e_p является единичным вектором, поэтому можно ожидать, что вектор ρ будет более разреженным, чем вектор π. Это, во-первых, может увеличить эффективность выполнения операции BTRAN (что, конечно, зависит от используемой факторизации базисной матрицы). И во-вторых, разреженность вектора ρ можно использовать для более эффективного вычисления строки ξ_p. Последнее обстоятельство обусловлено тем, что произведение матрицы на вектор можно вычислять двумя способами: как скалярные произведения строк матрицы на заданный вектор и как линейную комбинацию столбцов матрицы. Первый способ не позволяет учесть разреженность заданного вектора, но при использовании второго способа можно пропускать те столбцы матрицы, которым соответствуют нулевые элементы вектора. Применительно к формуле (2.28), где столбцами N^T являются строки матрицы N, второй способ означает, что вектор ξ_p вычисляется как линейная комбинация строк матрицы N:

$$\xi_p = -N^T\rho = -\sum_{i=1}^{m} \rho_i\,(i\text{-я строка матрицы } N). \tag{2.30}$$

[16] Это связано с тем, что p-ю строку симплекс-таблицы можно рассматривать как строку относительных оценок небазисных переменных для целевой функции, где коэффициент при базисной переменной $(x_B)_p$ равен единице, а коэффициенты при всех остальных переменных равны нулю.

Эффективное использование формулы (2.30) возможно лишь в том случае, когда матрица N доступна по строкам. Поэтому возникает техническая проблема, так как стандартный вариант модифицированного симплекс-метода предполагает хранение в явном виде лишь матрицы коэффициентов ограничений A в разреженном столбцовом формате (а также базисной матрицы B в факторизованном виде).

Для решения указанной проблемы Р. Биксби (R. E. Bixby) [7] предлагает хранить матрицу N в разреженном строчном формате. Заметим, что матрица N изменяется на каждой итерации, поэтому всякий раз после перехода к смежному базису старый столбец N_q, соответствующий выбранной небазисной переменной $(x_N)_q$, следует заменять новым столбцом A_k, соответствующим выбранной базисной переменной $(x_B)_p = x_k$. Так как N хранится по строкам, то для этого сначала нужно пройти по ненулевым элементам старого столбца, который доступен как соответствующий столбец матрицы A, удалить эти элементы из строк N, после чего пройти по ненулевым элементам нового столбца A_k и добавить эти элементы в строки N. По всей видимости технику Биксби имеет смысл применять в тех случаях, когда большинство столбцов матрицы A являются достаточно разреженными (что является типичным для большинства реальных ЛП-задач).

В качестве компромиссного варианта вместо хранения матрицы N можно хранить дубликат матрицы A в разреженном строчном формате. Это дает преимущество по сравнению с техникой Биксби, поскольку матрица A остается неизменной при переходе к смежному базису. Однако в этом случае непосредственное использование формулы (2.30) оказывается невозможным. Поэтому вместо вычисления вектора ξ_p как линейной комбинации строк матрицы N предлагается вычислять вектор α_p как линейную комбинацию строк матрицы A:

$$\alpha_p = -A^T\rho = -\sum_{i=1}^{m} \rho_i \,(i\text{-я строка матрицы } A). \tag{2.31}$$

Поскольку столбцы N являются подмножеством столбцов A, то используя соотношение (1.6) требуемый вектор ξ_p можно собрать из соответствующих элементов вектора α_p. Заметим, что при вычислении линейной комбинации α_p вычисляются «лишние» элементы (соответствующие столбцам матрицы B),[17] поэтому эффективность формулы (2.31) снижается с ростом числа ненулевых элементов вектора ρ. В связи с этим использовать формулу (2.31) предлагается лишь в тех случаях, когда заполнение ρ не очень велико (скажем, не более 10%), а в остальных случаях вычислять элементы вектора ξ_p как обычные скалярные произведения столбцов N на вектор ρ.

[17] Этими «лишними» элементами будут, очевидно, элементы единичного вектора e_p, так как из формулы (2.29) следует, что $B^T\rho = e_p$.

3 Выбор небазисной переменной

В соответствии с принципиальной схемой симплекс-метода, рассмотренной в конце подразд. 1.3, на шаге 2 требуется выбрать подходящую небазисную переменную $(x_N)_q$, изменение которой в допустимом направлении приводит к улучшению (уменьшению) целевой функции.[18] С геометрической точки зрения выбор небазисной переменной соответствует выбору ребра многогранного множества допустимых решений ЛП-задачи, исходящего из текущей вершины, вдоль которого имеет место улучшение целевой функции.

Пусть $J \subseteq \{1, \ldots, n-m\}$ — множество индексов небазисных переменных $(x_N)_j$, относительная оценка d_j которых имеет «неправильный» знак. Если $J \neq \varnothing$, то текущее базисное решение не является оптимальным, и в этом случае в качестве небазисной переменной можно, в принципе, выбрать любую переменную из J. Таким образом, проблема выбора состоит в принятии решения, какую именно небазисную переменную следует выбирать на каждой итерации симплекс-метода.

В данном разделе рассмотрены основные стратегии выбора небазисной переменной, получившие практическое применение.

3.1 Правило Данцига

Правило Данцига [2] состоит в выборе подходящей небазисной переменной с максимальной по абсолютной величине относительной оценкой, т. е. переменной $(x_N)_q$, для которой

$$|d_q| = \max_{j \in J} |d_j|. \tag{3.1}$$

Поскольку при изменении небазисной переменной $(x_N)_j$ на величину s целевая функция изменяется на величину $d_j s$, то правило Данцига обеспечивает наибольшее относительное улучшение целевой функции.

Основным преимуществом правила Данцига является его простота, поэтому оно является традиционным для табличного симплекс-метода. Однако фактическое изменение целевой функции при переходе к смежному базису определяется не только относительной оценкой выбранной небазисной переменной, но и фактическим изменением этой переменной, т. е. величиной $\theta_{\min}$ (1.23), которая зависит от выбранной небазисной переменной. При этом, как показывает практика, использование правила Данцига при решении реальных ЛП-задач во многих случях приводит к существенно бóльшему числу итераций симплекс-метода, необходимых для достижения оптимальной точки, по сравнению с другими, менее тривиальными стратегиями выбора небазисной переменной.

3.2 Метод наиболее крутого ребра

В своей статье [8] П. Харрис (P. M. J. Harris) заметила, что возможной причиной слишком медленной сходимости симплекс-метода при использовании правила Данцига является его асимметрия. Действительно, это правило основано на использовании относительных оценок d_j. Однако оценка, которая является максимальной по абсолютной величине с точки зрения текущего базиса уже может не быть таковой с точки зрения смежного базиса, поскольку

[18] Поскольку небазисные переменные соответствуют столбцам текущей симплекс-таблицы (1.8), то данную операцию часто называют *выбором столбца*.

симплекс-таблица для смежного базиса получается в результате линейного преобразования,[19] которое не является ортогональным, а значит, не сохраняет расстояния и углы.

Для устранения указанной асимметрии, Д. Гольдфарб (D. Goldfarb) и Дж. Рид (J. K. Reid) предложили использовать *метод наиболее крутого ребра (steepest edge)* [9]. В соответствии с этим методом предлагается выбирать ту подходящую небазисную переменную, для которой направление, определяемое соответствующим ребром, образует наиболее острый угол с направлением антиградиента (в случае минимизации) или градиента (в случае максимизации) целевой функции.

Отличительная особенность метода наиболее крутого ребра состоит в том, что указанный угол измеряется в пространстве всех (базисных и небазисных) переменных, а значит, не зависит от текущего базиса, поскольку пространство всех переменных остается одним и тем же на протяжении всех итераций. В этом смысле правило Данцига соответствует выбору ребра многогранника, которое также образует наиболее острый угол с антиградиентом (или градиентом в случае максимизации) целевой функции, но только в *текущем подпространстве небазисных переменных*, которое зависит от текущего базиса и, следовательно, изменяется на каждой итерации, что приводит к появлению асимметрии.

Рассмотрим вывод соответствующих формул.

Опуская перестановочную матрицу Π в формуле (1.11), запишем текущее базисное решение в виде:

$$x^0 = (x_B; x_N) = [(x_B)_1, \ldots, (x_B)_m; (x_N)_1, \ldots, (x_N)_{n-m}].$$

Геометрически точка x^0 является текущей вершиной многогранного множества в пространстве *всех* переменных ЛП-задачи. Допустим теперь, что некоторая небазисная переменная $(x_N)_j$ получила единичное приращение, причем все остальные небазисные переменные остались неизменными. Из (1.8) следует, что в результате получится следующая точка:

$$x^j = [(x_B)_1 + \xi_{1j}, \ldots, (x_B)_m + \xi_{mj}; (x_N)_1, \ldots, (x_N)_j + 1, \ldots, (x_N)_{n-m}],$$

где ξ_{1j}, $\ldots$, ξ_{mj} — элементы j-го столбца текущей симплекс-таблицы Ξ. Поскольку указанная смещенная точка лежит на прямой, проходящей через ребро многогранного множества, соответствующего небазисной переменной $(x_N)_j$, то направление этого ребра совпадает с направлением вектора-приращения:

$$\Delta x^j = x^j - x^0 = (\xi_{1j}, \ldots, \xi_{mj}; 0, \ldots, 1, \ldots, 0), \tag{3.2}$$

где единица расположена в позиции $(x_N)_j$.

Нас интересует угол φ_j между направлением ребра многогранного множества Δx^j и градиентом целевой функции ∇z:

$$\cos\varphi_j = \frac{(\Delta x^j)^T \nabla z}{\|\Delta x^j\| \|\nabla z\|}. \tag{3.3}$$

Для целевой функции имеем $z = c^T x = c_B x_B + c_N x_N$, откуда:

$$\nabla z = c = [(c_B)_1, \ldots, (c_B)_m; (c_N)_1, \ldots, (c_N)_{n-m}]. \tag{3.4}$$

Заметим, что из формул (1.15) и (1.16) с учетом определения (1.9) следует $d = c_N - N^T B^{-T} c_B = \Xi^T c_B + c_N$. Поэтому, используя формулы (3.3) и (3.4), имеем:

$$(\Delta x^j)^T \nabla z = \sum_{i=1}^{m} \xi_{ij} (c_B)_i + (c_N)_j = \Xi_j^T c_B + (c_N)_j = d_j, \tag{3.5}$$

[19] Это преобразование соответствует одному шагу исключения метода Гаусса—Жордана.

где Ξ_j — j-й столбец текущей симплекс-таблицы, d_j — относительная оценка небазисной переменной $(x_N)_j$. Подставляя далее выражение для скалярного произведения из (3.5) в (3.3) окончательно получим:

$$\cos\varphi_j = \frac{d_j}{\|\Delta x^j\|\|\nabla z\|}, \tag{3.6}$$

где

$$\|\Delta x^j\| = \left(\sum_{i=1}^{m} \xi_{ij}^2 + 1\right)^{1/2} = (\Xi_j^T \Xi_j + 1)^{1/2}, \tag{3.7}$$

$$\|\nabla z\| = \|c\| = (c^T c)^{1/2}. \tag{3.8}$$

Поскольку градиент целевой функции $\nabla z = c = \text{const}$ не зависит от текущего базиса, то для применения метода наиболее крутого ребра достаточно знать только относительные оценки d_j и нормы векторов-приращений $\|\Delta x^j\|$. При этом выбор наиболее подходящей небазисной переменной $(x_N)_q$ определяется следующим правилом (ср. с правилом Данцига из подразд 3.1):

$$\frac{|d_q|}{\|\Delta x^q\|} = \max_{j\in J} \frac{|d_j|}{\|\Delta x^j\|} = \max_{j\in J} \frac{|d_j|}{(\Xi_j^T \Xi_j + 1)^{1/2}}. \tag{3.9}$$

В случае табличного симплекс-метода вычисление норм $\|\Delta x^j\|$ не вызывает затруднений, поскольку симплекс-таблица $\Xi = \xi_{ij}$ доступна в явном виде на каждой итерации. Однако, если используется модифицированный симплекс-метод, вычисление отдельного столбца Ξ_j, необходимого для вычисления $\|\Delta x^j\|$, требует применения операции прямого преобразования (FTRAN). Поэтому прямое вычисление указанных норм для всех небазисных переменных хотя и возможно, но является неоправданно трудоемким, из-за чего такой способ не находит практического применения.

Чтобы избавиться от необходимости явного вычисления норм векторов-приращений для всех небазисных переменных на каждой итерации симплекс-метода, в работе [9] были предложены формулы пересчета этих норм при переходе к смежному базису (по аналогии с пересчетом компонент базисного решения из подразд. 2.3). Рассмотрим вывод этих формул.

Введем для удобства величины:

$$\gamma_j = \|\Delta x^j\|^2 = 1 + \Xi_j^T \Xi_j = 1 + \sum_{i=1}^{m} \xi_{ij}^2,\ j = 1, \ldots, n-m, \tag{3.10}$$

и допустим, что для текущего базиса эти величины уже известны. Допустим также, что в смежном базисе переменная $(x_B)_p$ становится небазисной, а $(x_N)_q$ — базисной, т. е. ведущим элементом симплекс-таблицы является ξ_{pq}. Нас интересуют формулы пересчета величин γ_j для смежного базиса.

Рассмотрим вначале пересчет величины γ_q для ведущего столбца ($j = q$). Используя формулы пересчета элементов симплекс-таблицы (1.33)—(1.36) из подразд. 1.4, имеем:

$$\overline{\gamma}_q = 1 + \sum_i \overline{\xi}_{iq}^2 = 1 + \overline{\xi}_{pq}^2 + \sum_{i\neq p} \overline{\xi}_{iq}^2 = 1 + \left(\frac{1}{\xi_{pq}}\right)^2 + \sum_{i\neq p} \left(\frac{\xi_{iq}}{\xi_{pq}}\right)^2.$$

Заметим, что:

$$\sum_i \left(\frac{\xi_{iq}}{\xi_{pq}}\right)^2 = \left(\frac{\xi_{pq}}{\xi_{pq}}\right)^2 + \sum_{i\neq p} \left(\frac{\xi_{iq}}{\xi_{pq}}\right)^2 = 1 + \sum_{i\neq p} \left(\frac{\xi_{iq}}{\xi_{pq}}\right)^2,$$

поэтому:

$$\overline{\gamma}_q = \left(\frac{1}{\xi_{pq}}\right)^2 + \sum_i \left(\frac{\xi_{iq}}{\xi_{pq}}\right)^2 = \frac{1}{\xi_{pq}^2}\left(1 + \sum_i \xi_{iq}^2\right),$$

откуда окончательно следует, что:

$$\overline{\gamma}_q = \frac{1}{\xi_{pq}^2}\gamma_q. \tag{3.11}$$

Рассмотрим теперь пересчет величин γ_j для остальных столбцов ($j \neq q$). Снова используя формулы пересчета элементов симплекс-таблицы (1.33)—(1.36) из подразд. 1.4, имеем:

$$\overline{\gamma}_j = 1 + \sum_i \overline{\xi}_{ij}^2 = 1 + \overline{\xi}_{pj}^2 + \sum_{i\neq p} \overline{\xi}_{ij}^2 = 1 + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2 + \sum_{i\neq p}\left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2.$$

Заметим, что:

$$\sum_i \left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2 = \sum_{i\neq p}\left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2 + \left(\xi_{pj} - \frac{\xi_{pq}\xi_{pj}}{\xi_{pq}}\right)^2 =$$
$$= \sum_{i\neq p}\left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2,$$

поэтому:

$$\overline{\gamma}_j = 1 + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2 + \sum_i \left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2 =$$
$$= 1 + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2 + \sum_i \xi_{ij}^2 + \sum_i \left(\frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2 - 2\sum_i \frac{\xi_{ij}\xi_{iq}\xi_{pj}}{\xi_{pq}} =$$
$$= \left(1 + \sum_i \xi_{ij}^2\right) + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2\left(1 + \sum_i \xi_{iq}^2\right) - 2\left(\frac{\xi_{pj}}{\xi_{pq}}\right)\sum_i \xi_{ij}\xi_{iq}.$$

откуда:

$$\overline{\gamma}_j = \gamma_j + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2 \gamma_q - 2\left(\frac{\xi_{pj}}{\xi_{pq}}\right)\sum_i \xi_{ij}\xi_{pq}.$$

Сумма $\sum_i \xi_{ij}\xi_{pq}$ представляет собой скалярное произведение j-го столбца симплекс-таблицы:

$$\Xi_j = \Xi e_j = B^{-1}Ne_j = -B^{-1}N_j$$

и ее q-го (ведущего) столбца Ξ_q, поэтому:

$$\sum_i \xi_{ij}\xi_{pq} = \Xi_j^T \Xi_q = (-B^{-1}N_j)\Xi_q = -N_j^T B^{-T}\Xi_q.$$

Таким образом, окончательно получим:

$$\overline{\gamma}_j = \gamma_j + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2 \gamma_q + 2\left(\frac{\xi_{pj}}{\xi_{pq}}\right) N_j^T B^{-T}\Xi_q. \tag{3.12}$$

Использование формулы пересчета (3.12) требует двух операций обратного преобразования (BTRAN) на каждой итерации модифицированного симплекс-метода: одна операция нужна для вычисления элементов ведущей строки симплекс-таблицы $\xi_p = (\xi_{pj})$,[20] а вторая — для вычисления вектора $B^{-T}\Xi_q$.

Следует отметить, что многократный пересчет величин γ_j вследствие ошибок округления может привести к значительному отклонению этих величин от их точных значений. Чтобы уменьшить влияние ошибок округления, в работе [3] предлагается, во-первых, использовать в формуле (3.11) не текущее значение γ_q с предыдущей итерации, а более точное значение этой величины, вычисленное непосредственно по формуле (3.10):

$$\gamma_q = 1 + \sum_i \xi_{iq}^2, \tag{3.13}$$

что не вызывает затруднений, поскольку в случае модифицированного симплекс-метода ведущий столбец текущей симплекс-таблицы $\Xi_q = (\xi_{iq})$ доступен в явном виде. Во-вторых, можно заметить, что:

$$\overline{\gamma}_j = 1 + \sum_i \overline{\xi}_{ij}^2 = 1 + \overline{\xi}_{pj}^2 + \sum_{i \neq p} \overline{\xi}_{ij}^2 \geq 1 + \overline{\xi}_{pj}^2 = 1 + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2. \tag{3.14}$$

Поэтому, если вследствие ошибок округления значение $\overline{\gamma}_q$, вычисленное по формуле (3.12), оказывается меньше, чем правая часть неравенства (3.14), то в качестве более точного значения $\overline{\gamma}_q$ предлагается использовать величину в правой части указанного неравенства.

Допустим, что p-я (ведущая) строка $\xi_p = (\xi_{pj})$ и q-й (ведущий) столбец $\Xi_q = (\xi_{iq})$ текущей симплекс-таблицы уже вычислены. Тогда практическая схема пересчета величин γ_j для смежного базиса может быть следующей.

1. Вычислить более точное значение γ_q для текущего базиса:

$$\gamma_q = 1 + \sum_{i=1}^{m} \xi_{iq}^2.$$

2. Вычислить вспомогательный вектор, используя операцию BTRAN:

$$u = B^{-T}\Xi_q.$$

3. Выполнить пп. 4—6 для $j = 1, \ldots, q-1, q+1, \ldots, n-m$.

4. Вычислить вспомогательную величину:

$$r_j = \xi_{pj}/\xi_{pq}.$$

5. Вычислить вспомогательное скалярное произведение:

$$s_j = N_j^T u,$$

где N_j — j-й столбец матрицы N, т. е. столбец матрицы коэффициентов ограничений A, соответствующий небазисной переменной $(x_N)_j$.

[20] Заметим, что эта строка также используется для пересчета относительных оценок небазисных переменных (см. подразд. 2.3).

6. Вычислить величину $\overline{\gamma}_j$ для смежного базиса:

$$\overline{\gamma}_j = \max(\gamma_j + r_j^2\gamma_q + 2r_j s_j,\ 1 + r_j^2).$$

7. Вычислить величину $\overline{\gamma}_q$ для смежного базиса:

$$\overline{\gamma}_q = \gamma_q/\xi_{pq}^2.$$ ■

Последнее замечание касается использования правила (3.9). Чтобы избежать извлечения квадратного корня для выбора небазисной переменной $(x_N)_q$ по методу наиболее крутого ребра, можно использовать эквивалентное правило:

$$\frac{d_q^2}{\gamma_q} = \max_{j\in J} \frac{d_j^2}{\gamma_j}. \tag{3.15}$$

Практическое решение задач линейного программирования с использованием метода наиболее крутого ребра в большинстве случаев приводит к существенному сокращению числа итераций симплекс-метода по сравнению со стандартным правилом выбора небазисных переменных, и как следствие этого, к уменьшению общего времени решения даже с учетом того, что на каждой итерации приходиться выполнять дополнительные вычисления.

Однако метод наиболее крутого ребра имеет один существенный практический недостаток. Как уже было отмечено выше, многократный пересчет величин γ_j на каждой итерации симплекс-метода может привести к значительному отклонению этих величин от их точных значений вследствие ошибок округления. Поэтому в какой-то момент может возникнуть необходимость прямого вычисления этих величин для текущего базиса непосредственно по формуле (3.10), для чего требуется вычислить *все* столбцы текущей симплекс-таблицы. С практической точки зрения это является неприемлемым, так как в случае модифицированного симплекс-метода вычисление одного столбца симплекс-таблицы требует выполнения одной операции прямого преобразования (FTRAN).

Аналогичная ситуация возникает и перед выполнением самой первой итерации симплекс-метода, когда величины γ_j еще не известны. Использование формулы (3.10) не вызывает затруднений лишь в единственном случае, а именно, когда начальный базис полностью состоит из вспомогательных переменных ограничений[21] (т. е. когда все исходные структурные переменные являются небазисными), поскольку в этом случае $B = I$, так что $\Xi = -B^{-1}N = -N$. Однако стандартный начальный базис является в некотором смысле исключением, поскольку все современные пакеты линейного программирования используют те или иные процедуры для формирования менее тривиального начального базиса. То же самое относится и к повторной оптимизации, например, при использовании техники генерации столбцов, где начальным базисом служит оптимальный базис редуцированной задачи.

[21] Введение вспомогательных неотрицательных или неположительных переменных в ограничения задачи является стандартной техникой преобразования неравенств к виду равенств.

3.3 Проекционный метод наиболее крутого ребра

Проекционный метод наиболее крутого ребра является развитием метода наиболее крутого ребра, рассмотренного в предыдущем подразделе. Основная идея этого метода состоит в том, чтобы измерять углы φ_j (3.3) не в пространстве всех переменных E^n, а в некотором подпространстве $V \subseteq E^n$, называемом *эталонным*, которое также остается неизменным на протяжении достаточно большого числа итераций симплекс-метода. Другими словами, в этом методе вместо формулы (3.3) используется формула:

$$\cos\varphi_j = \frac{(\Delta x_V^j)^T \nabla z_V}{\|\Delta x_V^j\| \|\nabla z_V\|}, \tag{3.16}$$

где Δx_V^j и ∇z_V — проекции, соответственно, векторов Δx^j и ∇z на эталонное подпространство V. При этом с практической точки зрения наиболее подходящим эталонным подпространством является *координатное подпространство*, определяемое небазисными переменными на некоторой произвольно выбранной итерации симплекс-метода.

Примечание. Правомочность выбора подпространства небазисных переменных в качестве эталонного имеет следующее обоснование. При рассмотрении табличного симплекс-метода (см. подразд. 1.4) было отмечено, что преобразованная задача (1.25)—(1.28), имеющая место на отдельной итерации симплекс-метода, эквивалентна исходной ЛП-задаче (1.1)—(1.3). Если исключить базисные переменные из преобразованной задачи, то мы получим следующую *редуцированную* задачу:

$$z = d^T x_N + d_0 \to \min$$

$$l_B \le \Xi x_N + g \le u_B$$

$$l_N \le x_N \le u_N$$

Эта редуцированная задача, очевидно, также эквивалентна исходной задаче, но включает только небазисные переменные исходной задачи, которые можно рассматривать как обычные переменные редуцированной задачи. Таким образом, пространство всех переменных редуцированной задачи — это подпространство небазисных переменных исходной задачи, выбранных на соответствующей итерации симплекс-метода. Следовательно, метод наиболее крутого ребра в пространстве всех переменных применительно к редуцированной задаче эквивалентен проекционному методу наиболее крутого ребра в подпространстве небазисных переменных применительно к исходной задаче.

Рассмотрим вывод соответствующих формул, полагая, что вектор-приращение Δx^j имеет вид (3.2).

Пусть V — эталонное координатное подпространство переменных, которое первоначально (т. е. на некоторой произвольно выбранной итерации симплекс-метода) совпадает с подпространством небазисных переменных. Определим следующие вспомогательные величины для текущего базиса:

$$\eta_i = \begin{cases} 1, & \text{если } (x_B)_i \in V \\ 0, & \text{если } (x_B)_i \notin V \end{cases} \tag{3.17}$$

$$\delta_j = \begin{cases} 1, & \text{если } (x_N)_j \in V \\ 0, & \text{если } (x_N)_j \notin V \end{cases} \tag{3.18}$$

Опуская для простоты перестановочную матрицу Π в формуле (1.11), можно считать, что $\Delta x^j = (\Delta x^j_B; \Delta x^j_N)$ и $\nabla z = (\nabla z_B; \nabla z_N)$, где подвекторы Δx^j_B и ∇z_B соответствуют базисным переменным x_B, а подвекторы Δx^j_N и ∇z_N соответствуют небазисным переменным x_N. Поэтому проекции указанных векторов на подпространство V будут равны:

$$\Delta x^j_V = (\eta \Delta x^j_B; \delta \Delta x^j_N), \tag{3.19}$$

$$\nabla z_V = (\eta \nabla z_B; \delta \nabla z_N), \tag{3.20}$$

где $\eta = \mathrm{diag}(\eta_i)$, $\delta = \mathrm{diag}(\delta_j)$ — диагональные (0,1)-матрицы. (Другими словами, проекция вектора на подпространство V получается в результате замены нулями тех его координат, которые не входят в V.)

Обратимся вначале к скалярному произведению в числителе формулы (3.16). Используя соотношения (3.19) и (3.20), а также учитывая, что $\eta^T \eta = \eta$ и $\delta^T \delta = \delta$, получим:

$$\begin{aligned}(\Delta x^j_V)^T \nabla z_V &= (\Delta x^j_B)^T \eta^T \eta \nabla z_B + (\Delta x^j_N)^T \delta^T \delta \nabla z_B = \\ &= (\Delta x^j_B)^T \eta \nabla z_B + (\Delta x^j_N)^T \delta \nabla z_B = (\Delta x^j)^T \nabla z_V.\end{aligned} \tag{3.21}$$

В любом базисе исходная целевая функция (1.1) эквивалентна целевой функции преобразованной задачи (1.25).[22] Следовательно, при использовании формулы (3.16) мы вправе считать, что начиная с той итерации симплекс-метода, когда выполняется фиксация эталонного подпространства V, мы вместо исходной целевой функции $z = c^T x + c_0$ имеем дело с *эквивалентной* целевой функцией преобразованной задачи $z = d^T x_N + d_0$ для указанной начальной итерации, как если бы исходной была эта эквивалентная целевая функция. Поскольку же на этой начальной итерации $\delta = \mathrm{diag}(\delta_j) = I$ (единичная матрица), то $\nabla z = \nabla z_V = (0; d)$, откуда с учетом (3.5) следует, что

$$(\Delta x^j)^T \nabla z_V = (\Delta x^j)^T \nabla z = d_j. \tag{3.22}$$

Важно отметить, что относительные оценки $d_j = (\Delta x^j) \nabla z$ для исходной целевой функции $z = c^T x + c_0$ и $d_j = (\Delta x^j)^T \nabla z_V$ для эквивалентной целевой функции $z = d^T x_N + d_0$ (где d и x_N относятся к итерации, на которой происходит фиксация эталонного подпространства) будут одними и теми же на всех последующих итерациях симплекс-метода вследствие эквивалентности целевых функций. Поэтому для вычисления относительных оценок d_j, которые относятся к эквивалентной целевой функции, никакая замена коэффициентов исходной целевой функции не требуется.

Подстановка скалярного произведения из (3.21) в числитель формулы (3.16) с учетом (3.22) дает следующее выражение для интересующего нас угла (подчеркнем, что теперь мы рассматриваем угол между проекцией вектора-приращения $\|\Delta x^j_V\|$ и проекцией градиента $\|\nabla z_V\|$ *эквивалентной* целевой функции, что однако не меняет существа дела):

$$\cos \varphi_j = \frac{d_j}{\|\Delta x^j_V\| \|\nabla z_V\|}. \tag{3.23}$$

[22] Целевую функцию преобразованной задачи (1.25) можно получить вычитанием из исходной целевой функции (1.1) линейной комбинации ограничений-равенств (1.2), если в качестве коэффициентов этой линейной комбинации использовать симплексные множители π_i (1.16). Заметим, что коэффициенты отдельного ограничения-равенства определяют вектор, ортогональный к плоскому множеству (аффинному подпространству) точек, удовлетворяющих системе ограничений-равенств (1.2), поэтому линейная комбинация таких векторов также будет обладать этим свойством ортогональности. Отсюда следует, что градиенты эквивалентных целевых функций отличаются друг от друга лишь этой ортогональной составляющей.

Так как $\nabla z = \text{const}$ и эталонное подпространство V не зависит от текущего базиса (остается неизменным), то $\nabla z_V = \text{const}$.[23] Поэтому для проекционного метода наиболее крутого ребра выбор наиболее подходящей небазисной переменной $(x_N)_q$ определяется следующим правилом:

$$\frac{|d_q|}{\|\Delta x_V^q\|} = \max_{j \in J} \frac{|d_j|}{\|\Delta x_V^j\|}, \tag{3.24}$$

которое является аналогом правила (3.9) для обычного метода наиболее крутого ребра (см. подразд. 3.2).

Перейдем теперь к нормам проекций векторов-приращений на эталонное подпространство. Из формул (3.2) и (3.19) следует, что

$$\Delta x_V^j = (\eta_1 \xi_{1j}, \ldots, \eta_m \xi_{mj}; 0, \ldots, \delta_j, \ldots, 0), \tag{3.25}$$

откуда:

$$\|\Delta x_V^j\| = \left(\sum_{i=1}^{m} \eta_i \xi_{ij}^2 + \delta_j \right)^{1/2} = \left(\sum_{i \in R} \xi_{ij}^2 + \delta_j \right)^{1/2}, \tag{3.26}$$

где $R = \{i : \eta_i = 1\} = \{i : (x_B)_i \in V\}$ — множество индексов базисных переменных, принадлежащих эталонному подпространству.

Введем для удобства величины:

$$\gamma_j = \|\Delta x_V^j\|^2 = \delta_j + \sum_{i \in R} \xi_{ij}^2, \ j = 1, \ldots, n - m, \tag{3.27}$$

которые аналогичны величинам (3.10) в обычном методе наиболее крутого ребра, и допустим, что для текущего базиса эти величины уже известны. Допустим также, что в смежном базисе переменная $(x_B)_p$ становится небазисной, а $(x_N)_q$ — базисной, т. е. ведущим элементом симплекс-таблицы является ξ_{pq}. Нас интересуют формулы пересчета величин γ_j для смежного базиса.

Рассмотрим вначале пересчет величины γ_q для ведущего столбца ($j = q$). В соответствии с (3.27) имеем:

$$\overline{\gamma}_q = \overline{\delta}_q + \sum_{i \in \overline{R}} \overline{\xi}_{iq}^2.$$

Так как в смежном базисе переменные $(x_B)_p$ и $(x_N)_q$ меняются местами, то $\overline{\eta}_p = \delta_q$, $\overline{\delta_q} = \eta_p$ и $\overline{R} \backslash \{p\} = R \backslash \{p\}$. Поэтому:

$$\overline{\gamma}_q = \overline{\delta}_q + \overline{\eta}_p \overline{\xi}_{pq}^2 + \sum_{i \in \overline{R} \backslash \{p\}} \overline{\xi}_{iq}^2 = \eta_p + \delta_q \overline{\xi}_{pq}^2 + \sum_{i \in R \backslash \{p\}} \overline{\xi}_{iq}^2.$$

Используя формулы пересчета (1.33)—(1.36), получим:

$$\overline{\gamma}_q = \eta_p + \delta_q \left(\frac{1}{\xi_{pq}} \right)^2 + \sum_{i \in R \backslash \{p\}} \left(\frac{\xi_{iq}}{\xi_{pq}} \right)^2.$$

[23] Очевидно, что $\nabla z = \nabla z_V$ в соответствии с определением данной эквивалентной целевой функции.

Заметим, что:

$$\sum_{i\in R}\left(\frac{\xi_{iq}}{\xi_{pq}}\right)^2 = \eta_p\left(\frac{\xi_{pq}}{\xi_{pq}}\right)^2 + \sum_{i\in R\setminus\{p\}}\left(\frac{\xi_{iq}}{\xi_{pq}}\right)^2 = \eta_p + \sum_{i\in R\setminus\{p\}}\left(\frac{\xi_{iq}}{\xi_{pq}}\right)^2,$$

поэтому:

$$\overline{\gamma}_q = \delta_q\left(\frac{1}{\xi_{pq}}\right)^2 + \sum_{i\in R}\left(\frac{\xi_{iq}}{\xi_{pq}}\right)^2 = \frac{1}{\xi_{pq}^2}\left(\delta_q + \sum_{i\in R}\xi_{iq}^2\right),$$

откуда следует, что:

$$\overline{\gamma_q} = \frac{1}{\xi_{pq}^2}\gamma_q. \tag{3.28}$$

Рассмотрим теперь пересчет величин γ_j для остальных столбцов ($j \neq q$). В соответствии с (3.27) имеем:

$$\overline{\gamma}_j = \overline{\delta}_j + \sum_{i\in\overline{R}}\overline{\xi}_{ij}^2.$$

Поскольку $j \neq q$, то $\overline{\delta}_j = \delta_j$. И, как уже было отмечено выше, $\overline{\eta}_p = \delta_q$ и $\overline{R}\setminus\{p\} = R\setminus\{p\}$. Поэтому:

$$\overline{\gamma}_j = \overline{\delta}_j + \overline{\eta}_p\overline{\xi}_{pj}^2 + \sum_{i\in\overline{R}\setminus\{p\}}\overline{\xi}_{ij}^2 = \delta_j + \delta_q\overline{\xi}_{pj}^2 + \sum_{i\in R\setminus\{p\}}\overline{\xi}_{ij}^2.$$

Используя формулы пересчета (1.33)—(1.36), получим:

$$\overline{\gamma}_j = \delta_j + \delta_q\left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2 + \sum_{i\in R\setminus\{p\}}\left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2.$$

Заметим, что:

$$\sum_{i\in R}\left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2 = \eta_p\left(\xi_{pj} - \frac{\xi_{pq}\xi_{pj}}{\xi_{pq}}\right)^2 + \sum_{i\in R\setminus\{p\}}\left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2$$
$$= \sum_{i\in R\setminus\{p\}}\left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2,$$

поэтому:

$$\overline{\gamma}_j = \delta_j + \delta_q\left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2 + \sum_{i\in R}\left(\xi_{ij} - \frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2 =$$
$$= \delta_j + \delta_q\left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2 + \sum_{i\in R}\xi_{ij}^2 + \sum_{i\in R}\left(\frac{\xi_{iq}\xi_{pj}}{\xi_{pq}}\right)^2 - 2\sum_{i\in R}\frac{\xi_{ij}\xi_{iq}\xi_{pj}}{\xi_p q} =$$
$$= \left(\delta_j + \sum_{i\in R}\xi_{ij}^2\right) + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2\left(\delta_q + \sum_{i\in R}\xi_{iq}^2\right) - 2\left(\frac{\xi_{pj}}{\xi_{pq}}\right)\sum_{i\in R}\xi_{ij}\xi_{iq},$$

откуда следует, что:

$$\overline{\gamma}_j = \gamma_j + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2\gamma_q - 2\left(\frac{\xi_{pj}}{\xi_{pq}}\right)\sum_{i\in R}\xi_{ij}\xi_{iq}.$$

Сумма $\sum_{i\in R}\xi_{ij}\xi_{iq} = \sum_{i=1}^{m}\xi_{ij}\eta_i\xi_{iq}$ представляет собой скалярное произведение j-го столбца симплекс-таблицы $\Xi_j = (\xi_{ij}) = -B^{-1}N_j$ и вектора $H\Xi_q$, где $\Xi_q = (\xi_{iq}) = -B^{-1}N_q$ — q-й (ведущий) столбец симплекс-таблицы, $H = \mathrm{diag}(\eta_i)$ — диагональная матрица, составленная из величин η_i (3.17). Поэтому:

$$\sum_{i\in R}\xi_{ij}\xi_{iq} = \Xi_j^T(H\Xi_q) = -N_j^T B^{-T} H\Xi_q.$$

Таким образом, окончательно получим:

$$\overline{\gamma}_j = \gamma_j + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2\gamma_q + 2\left(\frac{\xi_{pj}}{\xi_{pq}}\right)N_j^T B^{-T} H\Xi_q. \tag{3.29}$$

Можно видеть, что формулы пересчета (3.28) и (3.29) совпадают с аналогичными формулами (3.11) и (3.12) для обычного метода наиболее крутого ребра. Единственное отличие состоит в том, что в формуле (3.12) обратное преобразование применяется непосредственно к ведущему столбцу симплекс-таблицы Ξ_q, а в формуле (3.29) — к вектору $H\Xi_q$, который получается заменой некоторых элементов ведущего столбца нулями. Таким образом, оба метода имеют одинаковую вычислительную трудоемкость.

Как и в случае обычного метода наиболее крутого ребра, многократный пересчет величин γ_j по формулам (3.28) и (3.29) может привести к отклонению этих величин от их точных значений вследствие ошибок округления. Поэтому, чтобы уменьшить влияние таких ошибок, можно применять ту же технику, что и ранее. Так, в формуле (3.28) рекомендуется использовать не текущее значение γ_q с предыдущей итерации, а более точное значение этой величины, вычисленное непосредственно по формуле (3.27):

$$\gamma_q = \delta_q + \sum_{i\in R}\xi_{iq}^2. \tag{3.30}$$

Кроме того, можно заметить, что:

$$\begin{aligned}\overline{\gamma}_j = \overline{\delta}_j + \sum_{i\in\overline{R}}\overline{\xi}_{ij}^2 = \delta_j + \delta_q\overline{\xi}_{pj}^2 + \sum_{i\in R\setminus\{p\}}\overline{\xi}_{ij}^2 \geq \\ \geq \delta_j + \delta_q\overline{\xi}_{pj}^2 = \delta_j + \delta_q\left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2.\end{aligned} \tag{3.31}$$

Таким образом, если вследствие ошибок округления значение $\overline{\gamma}_j$, вычисленное по формуле (3.29), оказывается меньше, чем правая часть неравенства (3.31), то в качестве более точного значения $\overline{\gamma}_j$ имеет смысл использовать величину в правой части указанного неравенства.

Допустим, что p-я (ведущая) строка $\xi_p = (\xi_{pj})$ и q-й (ведущий) столбец $\Xi_q = (\xi_{iq})$ текущей симплекс-таблицы уже вычислены. Тогда практическая схема пересчета величин γ_j для смежного базиса в соответствии с проекционным методом наиболее крутого ребра может быть следующей.

1. Вычислить более точное значение γ_q для текущего базиса:

$$\gamma_q = \delta_q + \sum_{i\in R}\xi_{iq}^2.$$

2. Вычислить вспомогательный вектор, используя операцию BTRAN:

$$u = B^{-T}(H\Xi_q),$$

где $H = \mathrm{diag}(\eta_i)$ — диагональная матрица, составленная из величин η_i.

3. Выполнить пп. 4—6 для $j = 1, \ldots, q-1, q+1, \ldots, n-m$.

4. Вычислить вспомогательную величину:

$$r_j = \xi_{pj}/\xi_{pq}.$$

5. Вычислить вспомогательное скалярное произведение:

$$s_j = N_j^T u,$$

где N_j — j-й столбец матрицы N, т. е. столбец матрицы коэффициентов ограничений A, соответствующий небазисной переменной $(x_N)_j$.

6. Вычислить величину $\overline{\gamma}_j$ для смежного базиса:

$$\overline{\gamma}_j = \max(\gamma_j + r_j^2\gamma_q + 2r_j s_j,\ \delta_i + \delta_q r_j^2).$$

7. Вычислить величину $\overline{\gamma}_q$ для смежного базиса:

$$\overline{\gamma}_q = \gamma_q/\xi_{pq}^2.$$ ■

Последнее замечание касается использования правила (3.24). Чтобы не извлекать квадратный корень для вычислении нормы $\|\Delta_V^j\| = \gamma_j^{1/2}$ при выборе небазисной переменной $(x_N)_q$, можно использовать эквивалентное правило, которое совпадает с правилом (3.15) из подразд. 3.2:

$$\frac{d_q^2}{\gamma_q} = \max_{j \in J} \frac{d_j^2}{\gamma_j}. \tag{3.32}$$

Проекционный метод наиболее крутого ребра имеет те же положительные свойства, что и обычный метод наиболее крутого ребра в пространстве всех переменных. Однако возможность *переопределения* эталонного подпространства на любой итерации симплекс-метода позволяет избавиться от основного недостатка, присущего обычному методу наиболее крутого ребра и связанного с необходимостью вычисления всех столбцов симплекс-таблицы.

Начальное определение (инициализация) и переопределение эталонного подпространства возможны на любой итерации симплекс-метода и состоят в том, что эталонным подпространством становится текущее подпространство небазисных переменных. В соответствии с формулами (3.17) и (3.18) это означает, что $\eta_i = 0$ для всех $i = 1, \ldots, m$ (а значит, $R = \varnothing$) и $\delta_j = 1$ для всех $j = 1, \ldots, n-m$. Таким образом, как это следует из (3.27), сразу после инициализации или переопределения эталонного подпространства имеет место $\gamma_j = 1$ для всех $j = 1, \ldots, n-m$.

Поскольку проекционный метод наиболее крутого ребра основан на измерении углов в подпространстве, а не во всем пространстве переменных, то время от времени (например, через каждые 500 или 1000 итераций симплекс-метода) рекомендуется выполнять переопределение эталонного подпространства.

В заключение отметим, что проекционный метод наиболее крутого ребра по существу совпадает с методом оценивания Devex [8] (см. подразд. 3.4). Основное отличие состоит в том, что в данном методе весовые множители γ_j (3.27) пересчитываются точно, как это было предложено в статье [10] Х. Гринбергом (H. J. Greenberg) и Дж. Каланом (J. E. Kalan), а в методе Devex используется их достаточно грубая аппроксимация.

3.4 Метод оценивания Devex

Метод оценивания Devex — это метод выбора небазисной переменной, который был предложен П. Харрис (P. M. J. Harris) при разработке системы Devex [8]. Харрис первой обратила внимание на асимметрию правила Данцига и исходя из эмпирических соображений предложила измерять угол между выбираемым направлением и градиентом целевой функции не в текущем подпространстве небазисных переменных, которое изменяется на каждой итерации, а в фиксированном (эталонном) подпространстве.

Исторически метод оценивания Devex был первым методом выбора небазисной переменной, в котором используется идея эталонного (неизменного) подпространства. Этот метод по существу совпадает с проекционным методом наиболее крутого ребра (см. подразд. 3.3). Его единственное отличие состоит в том, что для пересчета масштабирующих величин γ_j (3.27)[24] Харрис использует приближенную формулу, которую можно получить из формулы (3.29), если отбросить последнее слагаемое:

$$\overline{\gamma}_j \approx \gamma_j + \left(\frac{\xi_{pj}}{\xi_{pq}}\right)^2 \gamma_q. \tag{3.33}$$

В данном случае требуется лишь одна операция обратного преобразования (BTRAN) на каждой итерации симплекс-метода для вычисления элементов ведущей строки симплекс-таблицы $\xi_p = (\xi_{pj})$,[25] поэтому использование метода оценивания Devex позволяет сэкономить одну операцию BTRAN на каждой итерации по сравнению с методами наиболее крутого ребра.

Допустим, что p-я (ведущая) строка $\xi_p = (\xi_{pj})$ и q-й (ведущий) столбец $\Xi_q = (\xi_{iq})$ текущей симплекс-таблицы уже вычислены. Тогда практическая схема пересчета величин γ_j для смежного базиса в соответствии с методом оценивания Devex может быть следующей. (Здесь считается, что эталонное подпространство V переменных определено посредством величин η_i (3.17) и δ_j (3.18) точно так же, как и в случае проекционного метода наиболее крутого ребра.)

1. Вычислить точное значение величины γ_q для текущего базиса:

$$\gamma_q = \delta_q + \sum_{i \in R} \xi_{iq}^2,$$

где $R = \{i : \eta_i = 1\} = \{i : (x_B)_i \in V\}$ — множество индексов базисных переменных, принадлежащих эталонному подпространству. (Заметим, что в соответствии с формулой (3.33) величины γ_j монотонно возрастают. Поэтому если вычисленное точное значение γ_q значительно отличается от его текущего приближенного значения, следует переопределить эталонное подпространство.)

2. Вычислить величины $\overline{\gamma}_j,\ j = 1, \ldots, q-1, q+1, \ldots, n-m$ для смежного базиса:

$$\overline{\gamma}_j = \gamma_j + (\xi_{pj}/\xi_{pq})^2 \gamma_q.$$

3. Вычислить величину $\overline{\gamma}_q$ для смежного базиса:

$$\overline{\gamma}_q = \gamma_q / \xi_{pq}^2.$$

■

[24] В статье [8] вместо величин γ_j используются сами нормы $T_j = \sqrt{\gamma_j} = \|\Delta x_V^j\|$, которые Харрис называет «tread» (англ. — ширина хода).

[25] Эта строка также используется для пересчета относительных оценок небазисных переменных (см. подразд. 2.3).

Выбор небазисной переменной $(x_N)_q$ выполняется по правилу (3.32) аналогично тому, как это делается при использовании проекционного метода наиболее крутого ребра. Начальное определение (инициализация) и переопределение эталонного подпространства также выполняются аналогичным образом.

4 Выбор базисной переменной

В соответствии с принципиальной схемой симплекс-метода, приведенной в конце подразд. 1.3, на шаге 3 требуется определить $\theta_{\min}$ — абсолютную величину приращения выбранной небазисной переменной $(x_N)_q$, и тем самым выбрать (шаг 4) базисную переменную $(x_B)_p$,[26] которая первой достигает своей границы при изменении $(x_N)_q$ в допустимом направлении. С геометрической точки зрения выбор базисной переменной представляет собой определение грани (гиперплоскости), соответствующей (нижней или верхней) границе базисной переменной, которая (грань) встречается первой при движении вдоль ребра многогранного множества допустимых решений ЛП-задачи, исходящего из текущей вершины и соответствующего выбранной небазисной переменной, и которая (грань), таким образом, определяет смежную вершину, где эта грань становится активным ограничением.

Для упрощения изложения всюду в данном разделе вместо границ переменных общего вида $l \le x \le u$ (1.3) рассматривается простейшее условие неотрицательности переменных $x \ge 0$ (т. е. случай, когда $l = 0$ и $u = +\infty$), что, однако, не влияет на общность получаемых результатов.[27]

В указанном простейшем случае активной границей выбранной небазисной переменной $(x_N)_q$ в текущем базисном решении может быть только ее нижняя нулевая граница, поэтому из (1.18) следует, что $s = +1$ и $(x_N)_q = \theta$, т. е. значение выбранной небазисной переменной совпадает с параметром луча, вдоль которого происходит движение. Зависимость (1.19) отдельной базисной переменной $(x_B)_i$ от θ в этом случае принимает вид:

$$(x_B)_i = \beta_i + \xi_{iq}\theta, \tag{4.1}$$

где $\beta_i \ge 0$ — значение $(x_B)_i$ в текущем базисном решении, ξ_{iq} — элемент симплекс-таблицы (1.9). Так как в рассматривамом случае базисные переменные также имеют нулевые (нижние) границы, то из (1.20) следует, что $(x_B)_i$ может достичь своей нулевой границы лишь убывая, т. е. когда $\xi_{iq} < 0$, при $\theta = \theta_i \ge 0$, где

$$\theta_i = -\frac{\beta_i}{\xi_{iq}}. \tag{4.2}$$

Соответственно, правило выбора (1.21) базисной переменной $(x_B)_p$ принимает вид:

$$\theta_{\min} = \theta_p = \min_{\xi_{iq}<0} \theta_i. \tag{4.3}$$

4.1 Проблема ведущего элемента

Ведущий элемент — это элемент ξ_{pq} текущей симплекс-таблицы (1.9), соответствующий выбранным базисной $(x_B)_p$ и небазисной $(x_N)_q$ переменным.

Заметим вначале, что из $\Xi_q = -B^{-1}N_q$ (2.16) следует:

$$B\Xi_q + N_q = 0, \tag{4.4}$$

[26] Поскольку базисные переменные соответствуют строкам текущей симплекс-таблицы (1.8), то данную операцию часто называют *выбором строки*.

[27] Так как ЛП-задачу (1.1)—(1.3) всегда можно преобразовать к эквивалентной ЛП-задаче с неотрицательными переменными.

или в развернутом виде:

$$\sum_{i=1}^{m} B_i \xi_{iq} + N_q = 0, \tag{4.5}$$

где B_i — i-й столбец текущей базисной матрицы, ξ_{iq} — i-й элемент q-го (ведущего) столбца текущей симплекс-таблицы Ξ_q, N_q — столбец матрицы коэффициентов ограничений, соответствующий выбранной небазисной переменной $(x_N)_q$. Базисная матрица для смежного базиса $\overline{B}$ получается из матрицы B заменой столбца B_p столбцом N_q. Поэтому если $\xi_{pq} = 0$, то из (4.5) следует, что в этом случае столбцы $\overline{B}$ будут линейно зависимыми, а значит, переход к смежному базису будет невозможен.

Логично ожидать, что чем ближе к нулю будет ведущий элемент, тем хуже будет обусловлена базисная матрица для смежного базиса, и тем хуже будет точность результатов операций FTRAN и BTRAN для такой базисной матрицы. Это подтверждается формулами пересчета элементов симплекс-таблицы при переходе к смежному базису (1.36) и (1.38) (см. подразд. 1.4): если ведущий элемент ξ_{pq} близок к нулю, то из ξ_{ij} и d_j будет вычитаться величины с большим порядком, что из-за ограниченной разрядности может привести к частичной или полной потере значащих цифр в исходных величинах ξ_{ij} и d_j.

Практика линейного программирования показывает, что при реализации симплекс-метода проблема ведущего элемента является главенствующей. Если абсолютная величина ведущего элемента оказывается неприемлемо малой, то обычно это приводит к существенной потере точности всех вычисляемых величин и, как следствие, к численной неустойчивости метода.

Здесь уместно отметить аналогичную проблему, которая возникает при реализации метода исключения Гаусса.[28] До тех пор, пока Уилкинсон (J. H. Wilkinson) в своей работе [11] не обратил внимание на фундаментальную роль выбора ведущего элемента, считалось, что метод исключения Гаусса является численно ненадежным и может применяться для практического решения систем лишь с очень малым (не более 10) числом уравнений и неизвестных. Это было связано с тем, что неудачный выбор ведущих элементов в процессе исключения обычно приводил к катастрофической потере точности и, как следствие, к неверным результатам.

В отличие от метода исключения Гаусса, где имеется достаточно большая свобода выбора ведущего элемента (например, среди элементов ведущего столбца или среди всех элементов активной подматрицы), выбор базисной переменной подчинен условию (4.3), которое практически сужает свободу выбора до единственного элемента, за исключением случаев, когда несколько базисных переменных $(x_B)_i$ одновременно, т. е. при одном и том же значении $\theta_i = \theta_{\min}$ (как правило, при $\theta_{\min} = 0$), достигают своей нулевой границы.

Поскольку выбор приемлемого по абсолютной величине ведущего элемента имеет принципиальное значение для численной устойчивости симплекс-метода, то должны быть предусмотрены меры, позволяющие увеличить свободу выбора. К таким мерам можно отнести:

ослабление условия (4.3) за счет разрешения незначительного (в пределах допуска) выхода значений базисных переменых за пределы своих границ;

ослабление границ переменных;

выбор другой подходящей небазисной переменной $(x_N)_q$.

[28] Пересчет симплекс-таблицы по формулам из подразд. 1.4 по существу представляет собой один шаг метода исключения Гаусса—Жордана.

(Вероятно, последняя возможность, аналогичная частичному выбору ведущего элемента в методе исключения Гаусса, является наиболее адекватной. Однако в случае модифицированного симплекс-метода ее использование затрудняется тем, что для этого может потребоваться просмотр большого числа столбцов симплекс-таблицы, а значит, выполнение большого числа операций FTRAN, необходимых для вычисления этих столбцов.)

4.2 «Стандартный» метод

«Стандартный» метод выбора базисной переменной (*textbook ratio test*) соответствует буквальному следованию формулам (4.2) и (4.3) (рис. 4.1).

```
«СТАНДАРТНЫЙ» МЕТОД
/* вначале ничего не выбрано */
p := 0, θmin := ∞;
/* основной цикл */
for i := 1, ..., m do
  if ξiq < 0 then
    /* (xB)i убывает при возрастании θ */
    θi := −βi/ξiq;
    /* (xB)i достигает нуля при θ = θi */
    if θmin > θi then
      p := i, θmin := θi;
    end if
  end if
end for
if p = 0 then STOP; /* ξiq ≥ 0 для всех i */
```

Рис. 4.1. «Стандартный» метод выбора базисной переменной.

В условиях приближенных вычислений использование «стандартного» метода выбора базисной переменной сопровождается следующими проблемами.

1. *Неприемлемый ведущий элемент.* Возможность выбора более подходящего ведущего элемента имеется только в случае, когда $\theta_i = \theta_{\min}$ для нескольких базисных переменных.

2. *Вырожденное решение.* Базисное решение является вырожденным, если для одной или нескольких базисных переменных имеет место $\theta_i = \theta_{\min} = 0$. Как уже было отмечено ранее, это замедляет сходимость и может привести к зацикливанию симплекс-метода.

3. *Нарушение границ.* Так как вектор значений базисных переменных в текущем базисном решении $\beta = (\beta_i)$ вычисляется приближенно, то при переходе к очередному смежному базису может случиться, что $\beta_i < 0$ для одной или нескольких базисных переменных, т. е. (нулевые) границы соответствующих базисных переменных могут оказаться нарушенными. Даже если нарушение границы является незначительным (в пределах допуска), нельзя сказать, нарушена ли граница в действительности или нет, так как точные значения β_i не известны. Однако в любом случае при использовании «стандартного» метода выбора базисной переменной будет иметь место $\theta_{\min} < 0$, т. е. в смежном базисе выбранная переменная $(x_N)_q$ изменится в недопустимом направлении (уменьшится), а значит, также будет нарушать свою (нулевую) границу.

УЛУЧШЕННЫЙ «СТАНДАРТНЫЙ» МЕТОД
/* вначале ничего не выбрано */
$p := 0,\ \theta_{\min} := \infty;$
/* основной цикл */
for $i := 1, \ldots, m$ **do**
  if $\beta_i < -\delta_1$ **then**
    STOP; /* чрезмерное нарушение границы */
  else if $\beta_i < +\delta$ **then**
    $\widetilde{\beta}_i := 0;$
  else
    $\widetilde{\beta}_i := \beta_i;$
  end if
  if $\xi_{iq} \le -\varepsilon$ /* $|\xi_{iq}| \ge \varepsilon$ */ **then**
    /* $(x_B)_i$ убывает при возрастании θ */
    $\theta_i := -\widetilde{\beta}_i / \xi_{iq};$
    /* $(x_B)_i$ достигает нуля при $\theta = \theta_i$ */
    if $\theta_{\min} > \theta_i$ OR ($\theta_{\min} = \theta_i$ AND $|\xi_{pq}| < |\xi_{iq}|$) **then**
      $p := i,\ \theta_{\min} := \theta_i;$
    end if
  end if
end for
if $p = 0$ **then** STOP; /* $\xi_{iq} > -\varepsilon$ для всех i */

Рис. 4.2. Улучшенный «стандартный» метод.

С учетом вышеизложенного «стандартный» метод можно несколько улучшить следующим образом (рис. 4.2).

1. Перед выполнением очередной итерации выполнять проверку, что $\beta_i \ge -\delta_1$ для всех базисных переменных, т. е. что текущее базисное решение является допустимым в пределах заданного допуска $\delta_1 > 0$ (например, для $\delta_1 = 10^{-8}$). Если же указанное условие нарушается, то следует прекратить поиск оптимального решения и перейти к поиску допустимого базисного решения для восстановления допустимости базиса.

2. Если $(-\delta_1 \le)\beta_i < 0$, то единственный выход в рамках «стандартного» симплекс-метода — считать, что точное значение β_i в этом случае равно нулю.

3. Если $\beta_i \le +\delta$ (например, для $\delta = 10^{-9}$), то по аналогии с предыдущим пунктом можно считать, что точное значение β_i в этом случае также равно нулю. Это позволяет искусственно увеличить вырожденность базисного решения и тем самым расширить свободу выбора ведущего элемента.

4. Если $|\xi_{iq}| < \varepsilon$ (например, для $\varepsilon = 10^{-10}$), т. е. если элемент симплекс-таблицы является очень малым по абсолютной величине, то считать, что в этом случае $\xi_{iq} = 0$, т. е. что базисная переменная $(x_B)_i$ не зависит от выбранной небазисной переменной $(x_N)_q$. (Заметим, что если точное значение ξ_{iq} на самом деле отлично от нуля, то пропуск $(x_B)_i$ может привести к нарушению границы этой переменной в смежном базисе.)

Использование улучшенного «стандартного» метода выбора базисной переменной позволяет частично разрешить проблему ведущего элемента, а также проблему нарушения границ

(если это нарушение не превышает заданного допуска). Тем не менее улучшенный метод имеет ряд недостатков.

Один из недостатков связан с выбором допуска ε на ведущий элемент. Задание слишком малого допуска может привести к выбору базисной переменной, для которой ведущий элемент является неприемлемо малым по абсолютной величине, а слишком большой допуск может привести к пропуску некоторых базисных переменных и, как следствие, к нарушению допустимости решения для смежного базиса.

Другой недостаток связан с выбором допуска δ для создания искусственной вырожденности базисного решения. Слишком малый допуск уменьшает свободу выбора, что может привести к выбору базисной переменной с неприемлемо малым по абсолютной величине ведущим элементом. Если же указанный допуск взять слишком большим и для выбранной базисной переменной $(x_B)_p$ будет иметь место $0 < \beta_p (\leq \delta)$, то фактическое ненулевое изменение параметра луча $\theta_{\min} = -\beta_p/\xi_{pq} > 0$ может привести к тому, что какая-либо другая базисная переменная $(x_B)_i$, которая имеет достаточно большой отрицательный коэффициент ξ_{iq} и для которой $\beta_i > \delta$ (вследствие чего эта переменная не может быть выбрана), получит слишком большое отрицательное приращение и поэтому нарушит свою нулевую границу в смежном базисном решении сверх разрешенного допуска.

4.3 Метод Харрис

Одна из основных проблем, которая возникает при выборе базисной переменной состоит в том, что выбранная базисная переменная $(x_B)_p$ имеет неприемлемо малый коэффициент (ведущий элемент) ξ_{pq}, однако выбор другой более подходящей базисной переменной «блокируется» нулевым или слишком близким к нулю значением β_p в текущем базисе. Примером может служить следующая ситуация:

$$(x_B)_1 = -1.0000000 \cdot \theta + 1.0$$
$$(x_B)_2 = -0.0000001 \cdot \theta + 0.0000000999$$

В данном случае $\theta_1 = 1$ и $\theta_2 = 0.999$, поэтому $\theta_{\min} = \theta_2$. Если следовать «стандартному» методу, то можно выбрать только $(x_B)_2$, хотя выбор $(x_B)_1$ был бы гораздо более предпочтительным с точки зрения величины ведущего элемента.

Практика линейного программирования показывает, что очень часто реальные ЛП-задачи имеют вырожденные или почти вырожденные базисные решения, где текущие значения β_i некоторых базисных переменных оказываются близкими к нулю. При этом указанные значения могут немного (в пределах допуска) нарушать нулевые границы соответствующих базисных переменных, что является нормальной ситуацией в условиях приближенных вычислений.

В своей статье [8] П. Харрис (P. M. J. Harris) предложила использовать указанное нарушение границ конструктивным образом, чтобы увеличить свободу выбора и тем самым обеспечить выбор базисной переменной с более подходящим ведущим элементом.

Применительно к рассмотренному выше примеру идея Харрис состоит в том, чтобы при достижении θ величины $\theta_2 = 0.999$, когда $(x_B)_2$ первой достигает своей нулевой границы, *продолжить* увеличение θ до величины $\theta_1 = 1$, когда своей нулевой границы достигнет $(x_B)_1$ с приемлемым коэффициентом. Понятно, что это приведет к выходу $(x_B)_2$ в недопустимую (отрицательную) область, однако, поскольку соответствующий коэффициент очень мал, то и нарушение границы тоже будет очень мало́ (а именно, $-0.0000001 + 0.0000000999 \approx 10^{-10}$).

МЕТОД ХАРРИС
/* первый просмотр */
$\widetilde{\theta}_{\min} := \infty$;
for $i := 1, \ldots, m$ **do**
  if $\beta_i < -\delta_1$ **then**
    STOP; /* чрезмерное нарушение границы */
  else if $\beta_i < 0$ **then**
    $\widetilde{\beta}_i := 0$;
  else
    $\widetilde{\beta}_i := \beta_i$;
  end if
  if $\xi_{iq} \le -\varepsilon$ /* $|\xi_{iq}| \ge \varepsilon$ */ **then**
    /* $(x_B)_i$ убывает при возрастании θ */
    $\widetilde{\theta}_i := -(\widetilde{\beta}_i + \delta)/\xi_{iq}$;
    /* $(x_B)_i$ достигает $(-\delta)$ при $\theta = \widetilde{\theta}_i$ */
    $\widetilde{\theta}_{\min} := \min(\widetilde{\theta}_{\min}, \widetilde{\theta}_i)$;
  end if
end for
if $\widetilde{\theta}_{\min} = \infty$ **then** STOP; /* $\xi_{iq} > -\varepsilon$ для всех i */
/* второй просмотр */
$p := 0$;
for $i := 1, \ldots, m$ **do**
  if $\xi_{iq} \le -\varepsilon$ /* $|\xi_{iq}| \ge \varepsilon$ */ **then**
    /* $(x_B)_i$ убывает при возрастании θ */
    $\theta_i := -\beta_i/\xi_{iq}$;
    /* $(x_B)_i$ достигает нуля при $\theta = \theta_i$ */
    if $\theta_i \le \widetilde{\theta}_{\min}$ AND ($p = 0$ OR $|\xi_{pq}| < |\xi_{iq}|$) **then**
      $p := i$;
    end if
  end if
end for
assert $p \ne 0$;

Рис. 4.3. Метод Харрис выбора базисной переменной.

Метод Харрис (рис. 4.3) состоит из двух этапов (просмотров списка базисных переменных). На первом этапе определяется величина $\widetilde{\theta}_{\min}$, которая, в отличие от «стандартного» метода (см. подразд. 4.2), определяется не для исходных нулевых границ базисных переменных $(x_B)_i \ge 0$, а для ослабленных границ $(x_B)_i \ge -\delta$, где $\delta > 0$ — заданный допуск на нарушение границ. Это означает, что для $\theta \le \widetilde{\theta}_{\min}$ выход значений базисных переменных за нулевую границу не может превысить заданного допуска δ. На втором этапе выполняется выбор базисной переменной $(x_B)_p$, для которой

$$|\xi_{pq}| = \max_{\xi_{iq}<0,\ \theta_i \le \widetilde{\theta}_{\min}} |\xi_{iq}|, \tag{4.6}$$

где $\theta_i = -\beta_i/\xi_{iq}$ — величина параметра луча θ, при котором переменная $(x_B)_i$ достигает своей исходной нулевой (не ослабленной) границы.

Понятно, что вследствие ослабления границ базисных переменных имеет место неравенство $\widetilde{\theta}_{\min} \ge \theta_{\min}$, где $\theta_{\min}$ — максимальное приращение параметра луча, которое было бы

определено «стандартным» методом для исходных нулевых (неослабленных) границ базисных переменных. Таким образом, в методе Харрис появляется дополнительная свобода, которую можно использовать для выбора базисной переменной с наиболее подходящим (максимальным по абсолютной величине) коэффициентом ξ_{pq} в соответствии с условием (4.6). При этом, поскольку $\theta_p \leq \widetilde{\theta}_{\min}$, выход значений базисных переменных за пределы их нулевых границ в смежном базисе не превысит величины допуска δ.

Идея Харрис основана на неявном предположении, что в текущем базисном решении базисные переменные не нарушают своих границ. Однако может случиться (хотя бы в результате применения самого метода Харрис), что (в пределах допуска) такое нарушение имеет место. В этом случае, т. е. если $(\delta_1 \leq)\beta_i < 0$ и $\xi_{iq} < 0$ хотя бы для одной базисной переменной $(x_B)_i$, Харрис [8] предлагает брать $\widetilde{\theta}_{\min} = 0$. С другой стороны, как уже было отмечено в предыдущем подразделе, небольшое нарушение границы может быть обусловлено ошибками округления, поэтому с практической точки зрения имеется больше оснований считать, что в случае $\beta_i < 0$ «точное» значение β_i равно нулю. Такое изменение по сравнению с оригинальным методом Харрис (отраженное на рис. 4.3) дает бóльшую свободу выбора, поскольку в этом случае всегда имеет место $\widetilde{\theta}_{\min} > 0$. При этом можно ожидать, что возможное дополнительное нарушение границ все же будет оставаться в пределах разрешенного допуска, тем более, что на последующих итерациях некоторые базисные переменные, нарушающие свои границы, могут возрасти и тем самым уменьшить общую степень нарушения границ.

В заключение отметим, что если метод Харрис приводит к выбору базисной переменной, отличной от той, которая была бы выбрана с использованием «стандартного» метода, то этот выбор всегда соответствует большему приращению параметра луча θ, что существенно с точки зрения предотвращения зацикливания симплекс-метода, и более подходящему (большему по абсолютной величине) ведущему элементу ξ_{pq}, что существенно с точки зрения численной устойчивости.

5 Отыскание начального допустимого базиса

В общем случае можно считать, что при решении ЛП-задачи имеется некоторый начальный базис, определяющий начальное базисное решение. Для применения симплекс-метода этот базис должен быть допустимым, поэтому если начальный базис не является допустимым, возникает проблема отыскания начального допустимого базиса.

Практический подход к решению указанной проблемы состоит в применении симплекс-метода для отыскания оптимального базиса некоторой *вспомогательной ЛП-задачи*, зная который можно построить допустимый базис исходной ЛП-задачи и затем использовать его в качестве начального. Такую технику обычно называют *двухэтапным симплекс-методом*, где первый этап соответствует решению вспомогательной задачи, а второй — решению основной (исходной) задачи.

Заметим, что формирование начального (не обязательно допустимого) базиса для первого этапа представляет собой самостоятельную проблему, которая здесь не рассматривается.

5.1 Ограничения в стандартном формате

Обратимся вначале к наиболее простому случаю, когда система ограничений ЛП-задачи имеет стандартный формат, т. е. вместо границ общего вида (1.3) заданы условия неотрицательности переменных. В этом случае допустимым базисным решением будет любое неотрицательное базисное решение системы равенств (1.2). Таким образом, нам требуется найти базисное решение, удовлетворяющее следующим условиям:

$$Ax = b \tag{5.1}$$

$$x \geq 0 \tag{5.2}$$

Рассмотрим вспомогательную ЛП-задачу, также представленную в стандартном формате:

$$\widetilde{z} = 1^T y = \sum_i y_i \to \min \tag{5.3}$$

$$Ax + y = b \tag{5.4}$$

$$x, y \geq 0 \tag{5.5}$$

где $y = (y_i)$ — вектор *вспомогательных* переменных, имеющих смысл *невязок*,[29] $\widetilde{z}$ — вспомогательная целевая функция, представляющая собой *сумму невязок*. Ключевым является следующее свойство данной вспомогательной ЛП-задачи. Предположим, что исходная система ограничений (5.1)—(5.2) имеет допустимое решение. Тогда, включая в это решение нулевые невязки y_i, мы получим оптимальное решение вспомогательной ЛП-задачи, поскольку сумма невязок $\widetilde{z}$, будучи суммой неотрицательных переменных, не может быть меньше нуля. И наоборот, если у нас имеется какое-либо оптимальное решение вспомогательной ЛП-задачи (5.3)—(5.5), в котором сумма невязок $\widetilde{z}$ равна нулю, то исключая из этого решения невязки y_i, которые будут равны нулю, так как их сумма равна нулю, мы получим допустимое решение исходной системы ограничений. Следовательно, множество допустимых решений исходной системы

[29] Так как i-е равенство вспомогательной задачи имеет вид $\sum_k a_{ik}x_k + y_i = b_i$, то y_i есть разность (невязка) между правой и левой частями соответствующего исходного равенства (5.1).

ограничений (5.1)—(5.2) совпадает с множеством оптимальных решений вспомогательной ЛП-задачи (5.3)—(5.5), в которых сумма невязок $\widetilde{z}$ равна нулю. Это означает, что для отыскания допустимого решения исходной системы ограничений достаточно найти оптимальное решение вспомогательной ЛП-задачи. Заметим также, что если минимальная сумма невязок $\widetilde{z}$ оказывается положительной, а не нулевой, то исходная система ограничений является несовместной (не имеет допустимых решений).

Вспомогательная задача (5.3)—(5.5) является задачей линейного программирования, поэтому для ее решения можно использовать симплекс-метод. Примечательно, что для данной вспомогательной задачи всегда можно указать начальный допустимый базис, поэтому проблема отыскания такого базиса не возникает. В этом начальном базисе $x_B = y$ (базисными являются вспомогательные переменные-невязки) и $x_N = x$ (небазисными являются исходные переменные). Поскольку небазисные переменные находятся на своих активных нулевых границах, то значения базисных переменных $y = (y_i)$ в таком базисном решении будут равны $b = (b_i)$. Без ограничения общности можно считать, что $b = (b_i) \geq 0$, т. е. что все правые части равенств (5.1) неотрицательны (в случае $b_i < 0$ достаточно умножить обе части i-го равенства на -1), поэтому указанное базисное решение будет допустимым, а значит, его можно использовать в качестве начального базиса при решении вспомогательной задачи симплекс-методом.

Предположим, что мы нашли оптимальное базисное решение вспомогательной ЛП-задачи (5.3)—(5.5), в котором сумма невязок $\widetilde{z}$, а значит, и все отдельные невязки y_i равны нулю. Чтобы перейти к допустимому базисному решению исходной системы ограничений (5.1)—(5.2) мы должны исключить из оптимального базисного решения вспомогательной ЛП-задачи все вспомогательные переменные y_i, поскольку их нет в исходной системе ограничений. В том случае, когда все y_i являются небазисными, проблем не возникает. Однако, если некоторая переменная y_i является базисной, хотя и равной нулю, ее нельзя просто исключить из базиса, поскольку тогда число базисных переменных станет меньше числа ограничений-равенств, что противоречит определению базисного решения (см. подразд. 1.2). Один из возможных способов состоит в том, чтобы предварительно «выгнать» вспомогательную переменную y_i из числа базисных, заменив ее на подходящую небазисную исходную переменную x_k.[30]

Перечислим основные практические недостатки рассмотренного метода отыскания начального базисного допустимого решения:

(a) система ограничений должна иметь стандартный формат;

(b) требуется включение дополнительных переменных y_i во вспомогательную ЛП-задачу;

(c) решение вспомогательной ЛП-задачи можно начинать только с предопределенного базиса, где $x_B = y$ и $x_N = x$, что не позволяет использовать какой-либо другой (заданный) начальный базис;

(d) возникает проблема преобразования оптимального базиса вспомогательной ЛП-задачи в допустимый базис исходной системы ограничений, если некоторые переменные y_i оказываются базисными.

[30]Такой переменной может быть исходная небазисная переменная, имеющая ненулевой элемент в текущей симплекс-таблице (1.9). Можно показать, что указанная переменная всегда найдется, если матрица A имеет полный строчный ранг. При этом в полученном смежном базисе значения переменных не изменятся, а значит, смежный базис также будет оптимальным.

Чтобы устранить недостатки (c) и (d), невязки можно отнести не к ограничениям-равенствам (5.1), а к условиям неотрицательности (5.2). В этом случае вспомогательная ЛП-задача будет следующей:

$$\widetilde{z} = -1^T y = -\sum_k y_k \to \min \tag{5.6}$$

$$Ax + Ay = b \tag{5.7}$$

$$x + y = 0 \tag{5.8}$$

$$x \geq 0,\ y \leq 0 \tag{5.9}$$

Не составляет труда убедиться в том, что, как и ранее, множество допустимых решений исходной системы ограничений (5.1)—(5.2) совпадает с множеством оптимальных решений данной вспомогательной ЛП-задачи, в которых сумма невязок $\widetilde{z}$, а значит, и все отдельные невязки y_k равны нулю.

Пусть дан некоторый произвольный (недопустимый) базис (x_B, x_N) для исходной системы ограничений (5.1)—(5.2). Этот базис всегда возможно преобразовать в соответствующий допустимый базис для вспомогательной ЛП-задачи (5.6)—(5.9) и затем использовать последний в качестве начального допустимого базиса для симплекс-метода. Чтобы сформировать этот начальный допустимый базис, мы вначале расширяем множество небазисных переменных x_N, включая в него все вспомогательные переменные y_k. В результате мы получим базис для вспомогательной ЛП-задачи, где все базисные переменные $(x_B)_i$ будут равны тем же значениям β_i, что и в базисе исходной системы ограничений, а все небазисные переменные будут равны нулю. Далее для каждого $\beta_i < 0$ мы выводим из базиса переменную $(x_B)_i = x_k$ и включаем вместо нее в базис соответствующую переменную y_k. Поскольку переменные x_k и y_k имеют одинаковые столбцы в матрице коэффициентов системы ограничений-равенств (5.7)—(5.8), то после указанной замены переменных матрицы B и N (1.6) не изменятся, а значит, значением базисной переменной y_k останется $\beta_i < 0$. Таким образом, после выполнения всех необходимых замен переменных мы получим базис, где базисной переменной $(x_B)_i$ является либо исходная переменная x_k (если $\beta_i \geq 0$), либо вспомогательная переменная y_k (если $\beta_i < 0$). Очевидно, что полученный базис будет допустимым для вспомогательной ЛП-задачи (5.6)—(5.9).

Рассмотренный второй метод также свободен от недостатка (d). Действительно, пусть мы получили оптимальное базисное решение вспомогательной ЛП-задачи (5.6)—(5.9), в котором сумма невязок $\widetilde{z}$, а значит, и все отдельные невязки y_k равны нулю. Если в этом оптимальном решении переменная y_k является базисной, то соответствующая переменная x_k не может быть базисной (иначе в базисной матрице B были бы два идентичных столбца, что невозможно, поскольку B — невырожденная матрица). Это позволяет вывести из числа базисных переменную y_k, заменяя ее переменной x_k. Как уже было отмечено выше, подобная замена переменных в базисе не изменяет значений базисных переменных, поэтому текущим значением x_k останется нуль.

Чтобы освободиться от недостатка (b), заметим, что во втором методе переменные x_k и y_k входят в ограничения (5.7)—(5.8) вспомогательной ЛП-задачи в виде суммы. Поскольку $x_k \geq 0$ и $y_k \leq 0$, причем эти переменные не могут быть базисными одновременно, их сумму можно заменить *одной* переменной:

$$\widetilde{x}_k = x_k + y_k, \tag{5.10}$$

значение которой однозначно определяет значения обеих переменных x_k и y_k в отдельности. Действительно, если $(x_B)_i = \widetilde{x}_k$, т. е. $\widetilde{x}_k$ является базисной, имея значение β_i, то случай $\beta_i \geq 0$ означает, что в базисе находится $x_k = (x_B)_i$, при этом y_k является небазисной (и равной нулю),

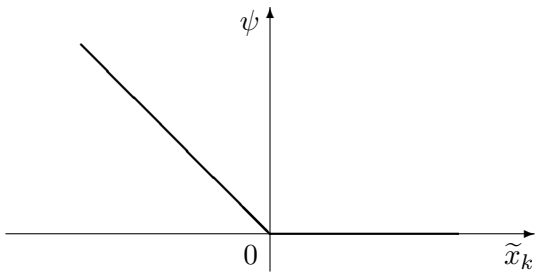

Рис. 5.1. Кусочно-линейная функция $\psi(\widetilde{x}_k)$.

а случай $\beta_i < 0$ означает, что в базисе находится $(x_B)_i = y_k$, при этом x_k является небазисной (и равной нулю). Если же $\widetilde{x}_k$ является небазисной, это означает, что обе переменные x_k и y_k являются небазисными (равными нулю).

Чтобы полностью избавиться от переменных x_k и y_k, достаточно представить сумму невязок (5.6) в виде:

$$\widetilde{z} = \sum_k \psi(\widetilde{x}_k), \tag{5.11}$$

где $\widetilde{x}_k$ — «комбинированная» переменная (5.10), $\psi(\widetilde{x}_k)$ — кусочно-линейная функция (рис. 5.1), определяемая следующим образом:

$$\psi(\widetilde{x}_k) = \begin{cases} 0, & \text{если } \widetilde{x}_k \geq 0 \\ -\widetilde{x}_k, & \text{если } \widetilde{x}_k < 0 \end{cases} \tag{5.12}$$

В итоге мы получим следующую вспомогательную задачу:

$$\widetilde{z} = \sum_k \psi(\widetilde{x}_k) \to \min \tag{5.13}$$

$$A\widetilde{x} = b \tag{5.14}$$

которая *эквивалентна* ЛП-задаче (5.6)—(5.9). Эта задача имеет *нелинейную* (кусочно-линейную) целевую функцию и поэтому уже не является задачей линейного программирования. Заметим также, что множество переменных этой задачи по существу совпадает с множеством переменных исходной системы ограничений (5.1)—(5.2), однако, условия неотрицательности (5.2) в данном случае отсутствуют, поскольку переменные $\widetilde{x}_k$, будучи суммой неотрицательной и неположительных переменных (5.10), могут иметь любой знак.

Функцию $\psi(\widetilde{x}_k)$ (5.12) можно рассматривать как «штраф» за нарушение условий неотрицательности (5.2) отдельной переменной $\widetilde{x}_k$. Поэтому целевая функция $\widetilde{z}$ (5.13) представляет собой так называемую *штрафную функцию*. На множестве допустимых решений исходной системы ограничений (5.1)—(5.2) эта функция равна нулю, а вне этого множества принимает положительные значения.[31]

Можно заметить, что $\psi(\widetilde{x}_k)$ является выпуклой кусочно-линейной функцией, откуда следует, что штрафная функция $\widetilde{z}$ есть выпуклая сепарабельная[32] кусочно-линейная функция. Множество допустимых решений вспомогательной задачи, т. е. множество точек, удовлетворяющих системе линейных ограничений-равенств (5.14), является плоским,[33] а значит, выпуклым. Вспомогательную задачу (5.13)—(5.14), таким образом, можно классифицировать как

[31] Таким образом, это *внешняя штрафная функция*, так как она определена на всем пространстве переменных.
[32] Имеющая вид суммы функций, каждая из которых зависит только от одной переменной.
[33] *Плоское множество* это другое название аффинного подпространства.

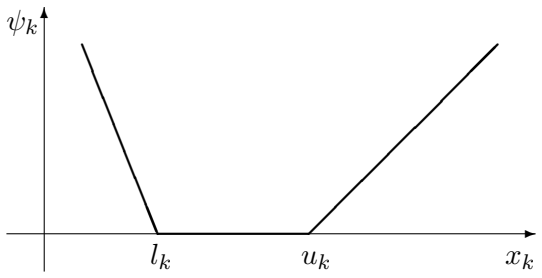

Рис. 5.2. Элементарная штрафная функция $\psi_k(x_k)$.

задачу выпуклого программирования с сепарабельной кусочно-линейной целевой функцией и линейными ограничениями-равенствами.

Техника решения вспомогательной задачи (5.13)—(5.14) здесь не рассматривается, поскольку в следующих подразделах рассмотрен более общий случай применительно к исходной системе ограничений в рабочем формате (1.2)—(1.3). Отметим однако, что применение симплекс-метода для решения указанной вспомогательной задачи выпуклого программирования есть на самом деле применение симплекс-метода для решения эквивалентной ЛП-задачи (5.6)—(5.9), где комбинированные переменные $\widetilde{x}_k$ одновременно играют роль как исходных переменных x_k, так и вспомогательных переменных y_k.

5.2 Вспомогательная задача для ограничений в общем формате

Рассмотрим ЛП-задачу в рабочем (общем) формате (1.1)—(1.3). В этом случае нам требуется найти базисное решение, удовлетворяющее следующим условиям:

$$Ax = b \tag{5.15}$$

$$l \le x \le u \tag{5.16}$$

Используя известную технику, указанную систему ограничений можно преобразовать к стандартному формату (5.1)—(5.2). Однако такой подход следует считать неудовлетворительным с практической точки зрения, так как в этом случае требуется включать в задачу дополнительные переменные и ограничения. Поэтому мы следуем подходу, рассмотренному в предыдущем подразделе, применяя его к системе ограничений в общем формате.

В данном случае отдельная исходная переменная x_k может иметь обе нижнюю l_k и верхнюю u_k границы. Поэтому в качестве элементарной штрафной функции вместо (5.12) мы используем аналогичную выпуклую кусочно-линейную функцию (рис. 5.2), определяемую следующим образом:

$$\psi_k(x_k) = \begin{cases} c_k^-(x_k - l_k), & \text{если } x_k < l_k \\ 0, & \text{если } l_k \le x_k \le u_k \\ c_k^+(x_k - u_k), & \text{если } x_k > u_k \end{cases} \tag{5.17}$$

где $c_k^- < 0$, $c_k^+ > 0$ — некоторые заданные коэффициенты, имеющие смысл весовых множителей (например, в простейшем случае можно считать, что $c_k^- = -1$ и $c_k^+ = +1$).

Понятно, что применительно к системе ограничений (5.15)—(5.16) вспомогательная задача с выпуклой кусочно-линейной сепарабельной штрафной функцией, которая аналогична задаче (5.13)—(5.14), будет иметь следующий вид:

$$z = \sum_k \psi_k(x_k) \to \min \tag{5.18}$$

$$Ax = b \tag{5.19}$$

Таким образом, для отыскания базисного допустимого решения для исходной системы ограничений (5.15)—(5.16) требуется решить указанную вспомогательную задачу. Если значение штрафной функции z в найденном решении равно нулю, что возможно только если все ψ_k равны нулю, значит, значения всех переменных x_k находятся внутри своих границ. Если же минимальное значение штрафной функции окажется положительным, то это будет означать, что исходная система ограничений несовместна (не имеет допустимых решений).

Вспомогательная задача (5.18)—(5.19) является задачей выпуклого программирования с нелинейной целевой функцией, поэтому непосредственное применение симплекс-метода для решения этой задачи невозможно. Чтобы перейти к линейной целевой функции достаточно заменить каждую переменную x_k во вспомогательной задаче суммой

$$x_k + y_k^- + y_k^+, \tag{5.20}$$

в результате чего получится следующая *эквивалентная ЛП-задача*:

$$z = (c^-)^T y^- + (c^+)^T y^+ \to \min \tag{5.21}$$

$$Ax + Ay^- + Ay^+ = b \tag{5.22}$$

$$l \le x \le u \tag{5.23}$$

$$y^- \le 0,\ y^+ \ge 0 \tag{5.24}$$

где $y^- = (y_k^-)$ и $y^+ = (y_k^+)$ — векторы, соответственно, *нижних* и *верхних* вспомогательных переменных, $c^- = (c_k^-)$ и $c^+ = (c_k^+)$ — векторы коэффициентов штрафной (целевой) функции при вспомогательных переменных, в роли которых (коэффициентов) выступают весовые множители соответствующих элементарных штрафных функций (см. выше).

Особенность применения симплекс-метода для решения ЛП-задачи (5.21)—(5.24) состоит в том, что вспомогательные переменные y_k^- и y_k^+ присутствуют в ЛП-задаче *неявно* — их роль выполняют исходные переменные x_k, поэтому можно считать, что симплекс-метод применяется непосредственно к вспомогательной задаче (5.18)—(5.19).

Возможность использования переменных x_k исходной системы ограничений (5.15)—(5.16) в качестве переменных y_k^- и y_k^+ вспомогательной ЛП-задачи (5.21)—(5.24) основана на том, что существует взаимно-однозначное соответствие между *всеми* (допустимыми и недопустимыми) базисными решениями исходной системы ограничений и *допустимыми* решениями вспомогательной ЛП-задачи. Действительно, пусть (x_B, x_N) — базисное решение исходной системы ограничений, для которого $\beta = (\beta_i)$ — вектор текущих значений базисных переменных. Если $(l_B)_i \le \beta_i \le (u_B)_i$, т. е. базисная переменная $(x_B)_i = x_k$ не нарушает своих границ, то в базисе вспомогательной ЛП-задачи x_k является базисной, а y_k^- и y_k^+ — небазисными и поэтому равными нулю. Если же $\beta_i < (l_B)_i$, т. е. базисная переменная $(x_B)_i = x_k$ нарушает свою нижнюю границу, то в базисе вспомогательной ЛП-задачи базисной является нижняя вспомогательная переменная y_k^-, переменная x_k является небазисной с активной нижней границей

$(l_B)_i = l_k$ и переменная y_k^+ является небазисной равной нулю. Заметим, что в этом случае значение базисной переменной y_k^- равно $\beta_i - (l_B)_i$ и представляет собой отрицательную величину нарушения нижней границы переменной $(x_B)_i = x_k$ с точки зрения исходной системы ограничений. Аналогично, если $\beta_i > (u_B)_i$, т. е. базисная переменная $(x_B)_i = x_k$ нарушает свою верхнюю границу, то в базисе вспомогательной ЛП-задачи базисной является верхняя вспомогательная y_k^+, переменная x_k является небазисной с активной верхней границей $(u_B)_i = x_k$ и переменная y_k^- является небазисной равной нулю. При этом значение базисной переменной y_k^+ равно $\beta_i - (u_B)_i$ и представляет собой положительную величину нарушения верхней границы переменной $(x_B)_i = x_k$ с точки зрения исходной системы ограничений.

5.3 Решение вспомогательной ЛП-задачи

Применение симплекс-метода для отыскания базисного допустимого решения исходной системы ограничений в общем формате (5.15)—(5.16) заключается в том, что мы решаем вспомогательную эквивалентную ЛП-задачу (5.21)—(5.24). Однако теперь решаемая ЛП-задача имеет неявное представление (чтобы избавиться от явного включения в исходную систему ограничений вспомогательных переменных y_k^- и y_k^+), поэтому процесс решения выглядит так, как если бы мы решали обычную ЛП-задачу в рабочем формате (1.1)—(1.3) с исходной системой системой ограничений (5.15)—(5.16), где специфика решаемой ЛП-задачи учитывается в самой схеме симплекс-метода (см. принципиальную схему симплекс-метода в конце подразд. 1.3).

Рассмотрим вначале произвольное (допустимое или недопустимое) базисное решение для исходной системы ограничений (5.15)—(5.16) с точки зрения эквивалентной ЛП-задачи (5.21)—(5.24). Пусть (x_B, x_N) — базис исходной системы ограничений, где $\beta = (\beta_i)$ — вектор текущих значений базисных переменных. Этот базис однозначно определяет соответствующий *допустимый* базис эквивалентной ЛП-задачи (см. предыдущий подраздел), где $(x_B)_i = x_k$ — базисная переменная исходной системы ограничений:

если $(l_B)_i \le \beta_i \le (u_B)_i$ (границы не нарушены), то x_k — базисная с текущим значением β_i, y_k^- и y_k^+ — небазисные с активными нулевыми границах;

если $\beta_i < (l_B)_i$ (нарушена нижняя граница), то y_k^- — базисная с текущим значением $\beta_i - (l_B)_i < 0$, x_k — небазисная с активной нижней границей, y_k^+ — небазисная с активной нулевой границей;

если $\beta_i > (u_B)_i$ (нарушена верхняя граница), то y_k^+ — базисная с текущим значением $\beta_i - (u_B)_i > 0$, x_k — небазисная с активной верхней границей, y_k^+ — небазисная с активной нулевой границей;

если $x_k = (x_N)_j$, т. е. x_k является небазисной переменной, то обе вспомогательные переменные y_k^- и y_k^+ также небазисные с активной нулевой границей.

Перейдем теперь к описанию принципиальной схемы симплекс-метода (ср. с подразд. 1.3), полагая, что мы решаем ЛП-задачу в рабочем формате (1.1)—(1.3), система ограничений которой совпадает с исходной системой ограничений (5.15)—(5.16), а целевая функция отражает свойства текущего базисного решения с точки зрения эквивалентной ЛП-задачи (5.21)—(5.24).

Шаг 1a. Вычислить β (1.12) — вектор текущих значений базисных переменных x_B.

Шаг 1b. Сформировать текущий вектор $c = (c_k)$ коэффициентов целевой функции, соответствующий вектору коэффициентов штрафной функции вспомогательной ЛП-задачи (5.21):

если $x_k = (x_B)_i$ и $\beta_i < (l_B)_i = l_k$, то с точки зрения эквивалентной ЛП-задачи в базисе находится нижняя вспомогательная переменная y_k^-, поэтому в соответствии с (5.21) установить $c_k = c_k^-$;

если $x_k = (x_B)_i$ и $\beta_i < (u_B)_i = u_k$, то с точки зрения эквивалентной ЛП-задачи в базисе находится верхняя вспомогательная переменная y_k^+, поэтому в соответствии с (5.21) установить $c_k = c_k^+$;

если $x_k = (x_B)_i$ и $(l_B)_i = l_k \le \beta_i \le (u_B)_i = u_k$, то с точки зрения ЛП-задачи в базисе находится сама исходная переменная x_k, поэтому в соответствии с (5.21) установить $c_k = 0$;

если же $x_k = (x_N)_j$, то с точки зрения эквивалентной ЛП-задачи все три переменные x_k, y_k^- и y_k^+ являются небазисными, поэтому установить $c_k = 0$, как если бы вспомогательные переменные y_k^- и y_k^+ были исключены из эквивалентной ЛП-задачи.

Можно отметить, что ненулевыми будут только коэффициенты при базисных переменных, нарушающих свои границы. При этом знак коэффициента c_k показывает какая именно граница переменной x_k нарушена:

$c_k < 0$ — нарушена нижняя граница;

$c_k > 0$ — нарушена верхняя граница;

$c_k = 0$ — нет нарушения границ.

Шаг 1c. Вычислить d (1.16) — вектор относительных оценок небазисных переменных x_N для текущего вектора коэффициентов целевой функции $c = (c_k)$.

С точки зрения эквивалентной ЛП-задачи d_j — это относительная оценка исходной переменной $x_k = (x_N)_j$, поскольку $c_k = 0$ (см. предыдущий шаг). Относительные оценки вспомогательных небазисных переменных y_k^- и y_k^+, как это следует из (1.15), будут равны соответственно $d_j + c_k^-$ и $d_j + c_k^+$.

Шаг 2. Выбрать подходящую небазисную переменную $(x_N)_q$, изменение которой в допустимом направлении уменьшает целевую функцию, т. е. относительная оценка d_q которой имеет «неправильный» знак. Небазисная переменная $(x_N)_j$ считается подходящей, если:

$d_j < 0$ и $(x_N)_j$ имеет активную нижнюю границу;

$d_j > 0$ и $(x_N)_j$ имеет активную верхнюю границу;

$d_j \ne 0$ и $(x_N)_j$ является свободной (неограниченной) переменной.

Фиксированные небазисные переменные не могут изменяться и поэтому не включаются в число подходящих. Небазисные вспомогательные переменные y_k^- или y_k^+ считаются исключенными из эквивалентной ЛП-задачи (зафиксированными в нуле),[34] поэтому они также не включаются в число подходящих. Таким образом, выбранной небазисной переменной $(x_N)_q$ может быть только некоторая исходная переменная x_k.

Если выбор невозможен (текущее базисное решение эквивалентной ЛП-задачи является оптимальным), то перейти к шагу 5.

[34]Так как в любом оптимальном решении эквивалетной ЛП-задачи с нулевым значением штрафной функции все вспомогательные переменные должны обратиться в нуль.

Шаг 3. Определить $\theta_{\min}$ — абсолютную величину приращения выбранной небазисной переменной $(x_N)_q$ в смежном базисе.

Как и в обычном случае (см. подразд. 1.3) $\theta_{\min}$ определяется по формуле (1.23). Однако при определении θ_i (точки, в которой базисная переменная $(x_B)_i$ достигает своей границы) по формуле (1.20) необходимо учитывать, что базисной переменной может быть как исходная переменная x_k, так и вспомогательная переменная y_k^- или y_k^+. Поэтому в формуле (1.20) вместо границ $(l_B)_i = l_k$ и $(u_B)_i = u_k$ следует использовать границы, определяемые следующим образом:

$$(l_B)'_i = \begin{cases} l_k, & \text{если } (x_B)_i = x_k \quad (c_k = 0) \\ -\infty, & \text{если } (x_B)_i = y_k^- \quad (c_k < 0) \\ u_k, & \text{если } (x_B)_i = y_k^+ \quad (c_k > 0) \end{cases}$$

$$(u_B)'_i = \begin{cases} u_k, & \text{если } (x_B)_i = x_k \quad (c_k = 0) \\ l_k, & \text{если } (x_B)_i = y_k^- \quad (c_k < 0) \\ +\infty, & \text{если } (x_B)_i = y_k^+ \quad (c_k > 0) \end{cases}$$

Смысл границ $(l_B)'_i$ и $(u_B)'_i$ можно пояснить следующим образом. Если $(x_B)_i = x_k$, то в текущем базисе переменная x_k не нарушает своих границ, поэтому используются ее оригинальные границы. Если $(x_B)_i = y_k^-$, то в текущем базисе x_k нарушает свою нижнюю границу. Следовательно, y_k^- возрастая достигнет своей верхней нулевой границы в той же точке θ_i, в которой возрастая достигла бы своей *нижней* границы исходная переменная x_k. Аналогично, если $(x_B)_i = y_k^+$, то в текущем базисе x_k нарушает свою верхнюю границу. Следовательно, y_k^+ убывая достигнет своей нижней нулевой границы в той же точке θ_i, в которой убывая достигла бы своей *верхней* границы исходная переменная x_k.

Заметим, что случай $\theta_{\min} = \infty$, соответствующий неограниченному минимуму штрафной функции, невозможен, поскольку штрафная функция по определению ограничена снизу нулем.

Шаг 4. Если $\theta_{\min} = \theta_p$ (1.21), то пометить переменную $(x_B)_p$ как небазисную, а переменную $(x_N)_q$ как базисную. Если же $\theta_{\min} = \theta_0$ (1.22), то изменить текущую активную границу переменной $(x_N)_q$ на противоположную.

Переменная $(x_N)_q$, которая становится базисной — это всегда некоторая исходная переменная (см. шаг 2). Однако переменная $(x_B)_p$, которая становится небазисной, может быть как исходной переменной x_k, так и вспомогательной переменной y_k^- или y_k^+. Так как в текущем базисе всегда находится ровно одна из этих трех переменных (другие две должны быть небазисными, поскольку все эти три переменные имеют идентичные столбцы в матрице коэффициентов ограничений), то в смежном базисе все эти три переменные будут небазисными. Поэтому если $(x_B)_p = x_k$, то в смежном базисе активная граница этой переменной определяется обычным образом (см. шаг 4 в подразд. 1.3). Если же $(x_B)_p = y_k^-$, то в смежном базисе x_k остается на своей активной нижней границе, а если $(x_B)_p - y_k^+$, то в смежном базисе x_k остается на своей активной верхней границы. Так как x_k используется для представления всех трех переменных, то последние два случая определяют активную границу x_k в смежном базисе.

Вернуться к шагу 1а.

Шаг 5. Пусть z^* — значение штрафной функции в текущем базисном решении, которое является оптимальным.

Если $z^* > 0$, то СТОП — исходная система ограничений (5.15)—(5.16) несовместна (не имеет допустимых решений).

Если $z^* = 0$, то с точки зрения вспомогательной ЛП-задачи все переменные y_k^- и y_k^+ в текущем базисном решении равны нулю. Если при этом y_k^- или y_k^+ является базисной, то хотя формально переменная x_k нарушает свою нижнюю или, соответственно, верхнюю границу, это нарушение равно нулю. Поэтому СТОП — текущий оптимальный базис эквивалентной ЛП-задачи является допустимым базисом исходной системы ограничений (5.15)—(5.16). ■

Примечание. Вместо условия оптимальности, проверяемого на шаге 2, в качестве условия останова можно использовать следующее более слабое условие. Поскольку в конечном итоге нам требуется допустимый базис исходной системы ограничений, то после вычисления текущих значений базисных переменных $\beta = (\beta_i)$ на шаге 1a достаточно проверить, что все исходные базисные переменные находятся внутри своих границ, т. е. что все *базисные* вспомогательные переменные y_k^- и y_k^+ равны нулю. Если это условие выполняется, то можно сразу перейти к шагу 5.

В заключение отметим, что при пересчете относительных оценок $d = (d_j)$ для смежного базиса (см. подразд. 2.3) необходимо иметь в виду следующее. Формула пересчета (2.25) получена исходя из условия, что коэффициенты целевой функции в смежном базисе не изменяются. Это условие выполняется, если с точки зрения эквивалентной ЛП-задачи из базиса выходит исходная переменная $(x_B)_p = x_k$, так как коэффициент штрафной функции c_k при этой переменной остается равным нулю в смежном базисе. Однако если из базиса выходит вспомогательная нижняя $(x_B)_p = y_k^-$ или верхняя $(x_B)_p = y_k^+$ переменная, то коэффициент штрафной функции c_k, который в текущем базисе относится к вспомогательной переменной и поэтому равен c_k^- или c_k^+, в смежном базисе будет относиться уже к исходной переменной x_k, а значит, будет равен нулю (см. шаг 1b). Поэтому после пересчета относительных оценок по формулам (2.25) и (2.26) в случае $(x_B)_p = y_k^-$ или $(x_B)_p = y_k^+$ необходимо скорректировать значение $\overline{d}_q$, вычитая из него, соответственно, c_k^- или c_k^+, как это следует из определений (2.13) и (2.14).

Литература

[1] G. B. Dantzig. Programming of Interdependent Activities, II, Mathematical Model, pp. 19-32, in: Activity Analysis of Production and Allocation, ed. T. C. Koopmans. New York: John Wiley & Sons, Inc., 1951. См. также: Econometrics 17, No. 3, 4 (July — Oct. 1949), pp. 200-211.

[2] G. B. Dantzig. Linear Programming and Extensions. Princeton, N. J.: Princeton University Press, 1963. [Имеется перевод: Дж. Данциг. Линейное программирование, его применения и обобщения.— М.: Прогресс, 1966.]

[3] G. B. Dantzig. Computational Algorithm of the Revised Simplex Method. RAND RM-1266, 1953.

[4] G. B. Dantzig, W. Orchard-Hays. Alternate Algorithm for the Revised Simplex Method. RAND RM-1268, 1953.

[5] G. B. Dantzig, A. Orden, P. Wolfe. The Generalized Simplex Method. RAND RM-1264, 1954.

[6] Mathematical Programming System/360 (360A-CO-14X). N. Y.: IBM Corp., 1966.

[7] R. E. Bixby. Progress in linear programming. ORSA J. on Computing, 6, 1994, pp. 15-22.

[8] P. M. J. Harris. Pivot Selection Methods of the Devex LP Code. Mathematical Programming 12, 1973, pp. 1-28.

[9] D. Goldfarb, J. K. Reid. A Practicable Steepest-Edge Simplex Algorithm. Mathematical Programming 12, 1977, pp. 361-371.

[10] H. J. Greenberg, J. E. Kalan. An exact update for Harris' TREAD. Mathematical Programming Study 4, 1975, pp. 26-29.

[11] J. H. Wilkinson. Error analysis of direct methods of matrix inversion. J. Assoc. Comp. Mach. 8, 1961, pp. 281-330.

Приложение

Программная реализация

П.1 Модуль SPXLP — основные операции

П.1.1 Структурный тип SPXLP

Структурный тип данных `SPXLP` используется для представления исходных данных ЛП-задачи в рабочем формате (1.1)—(1.3), текущего базиса, который определяется соответствующей перестановочной матрицей Π (1.4) и признаками активных границ небазисных переменных, а также факторизации базисной матрицы (1.6) для текущего базиса.

Матрица коэффициентов ограничений A имеет размерность $m \times n$ и хранится в разреженном столбцовом формате. Векторы хранятся в обычном плотном формате. Для представления факторизации базисной матрицы B, имеющей порядок m (по числу базисных переменных, совпадающему с числом ограничений-равенств), используется структурный тип данных `BFD` (драйвер факторизации базисной матрицы).

Структура `SPXLP` имеет следующие поля.

`int m` — число ограничений-равенств (1.2), число строк матрицы A, $m > 0$.

`int n` — число переменных $x = (x_k)$, число столбцов матрицы A, $n \geq m$.

`int nnz` — число ненулевых элементов в матрице A.

`int *A_ptr` — указатель на массив длины $1+n+1$. Ячейка `A_ptr[0]` не используется. Ячейка `A_ptr[k]`, $1 \leq k \leq n$, определяет начальную позицию k-го столбца матрицы A в массивах `A_ind` и `A_val`, при этом всегда `A_ptr`$[1] = 1$ (т. е. первый столбец начинается в первой позиции). `A_ptr`$[\mathtt{n}+1] = nnz + 1$ определяет позицию, следующую за последним элементом в массивах `A_ind` и `A_val` (т. е. позицию, в которой бы начинался $(n+1)$-й столбец). Столбцы матрицы следуют друг за другом без промежутков, поэтому длина (число ненулевых элементов) k-го столбца равна `A_ptr`$[\mathtt{k}+1]$ − `A_ptr`$[\mathtt{k}] \geq 0$.

`int *A_ind` — указатель на массив длины $1+$`nnz`. Ячейка `A_ind[0]` не используется. Ячейки `A_ind[1]`, ..., `A_ind[nnz]` содержат строчные индексы соответствующих элементов матрицы A.

`double *A_val` — указатель на массив длины $1+$`nnz`. Ячейка `A_val[0]` не используется. Ячейки `A_val[1]`, ..., `A_val[nnz]` содержат (ненулевые) значения соответствующих элементов матрицы A.

`double *b` — указатель на массив длины $1+m$. Ячейка `b[0]` не используется. Ячейка `b[i]`, $1 \leq i \leq m$, содержит значение правой части i-го ограничения-равенства b_i (1.2).

`double *c` — указатель на массив длины $1+n$. Ячейка `c[0]` содержит значения постоянного члена целевой функции c_0 (1.1). Ячейка `c[k]`, $1 \leq k \leq n$, содержит значение коэффициента целевой функции при переменной x_k.

`double *l` — указатель на массив длины $1+n$. Ячейка `l[0]` не используется. Ячейка `l[k]`, $1 \leq k \leq n$, содержит значение нижней границы l_k переменной x_k (1.3). Если $l_k = -\infty$ (x_k не имеет нижней границы), то `l`$[\mathtt{k}] = -$`DBL_MAX`.

`double *u` — указатель на массив длины $1+n$. Ячейка `u[0]` не используется. Ячейка `u[k]`, $1 \le k \le n$, содержит значение верхней границы u_k переменной u_k (1.3). Если $u_k = +\infty$ (x_k не имеет верхней границы), то `u[k]` $= +$`DBL_MAX`. Если x_k является фиксированной переменной, то `u[k]` $=$ `l[k]`.

`int *head` — указатель на массив длины $1 + n$, называемый *заголовком базиса*. Ячейка `head[0]` не используется. Ячейка `head[i]`, $1 \le i \le m$, содержит исходный номер k переменной, которая занимает i-ю позицию в векторе базисных переменных x_B, т. е. для которой $(x_B)_i = x_k$. Ячейка `head[m+j]`, $1 \le j \le n-m$, содержит исходный номер k переменной, которая занимает j-ю позицию в векторе небазисных перемнных x_N, т. е. для которой $(x_N)_j = x_k$. Таким образом, заголовок базиса задает перестановочную матрицу Π, которая определяет разбиение переменных (1.4) на базисные и небазисные в текущем базисе: `head[k']` $= k$ означает, что $\Pi[k', k] = 1$.

`char *flag` — указатель на массив длины $1 + n - m$. Ячейка `flag[0]` не используется. Ячейка `flag[j]`, $1 \le j \le n - m$, содержит признак активной верхней границы для небазисной переменной $(x_N)_j$. Этот признак может быть установлен только для небазисных переменных, которые имеют верхнюю границу и не являются фиксированными.

`int valid` — если данный признак установлен, то факторизация текущей базисной матрицы B является действительной. В противном случае, если данный признак сброшен, факторизация не действительна.

`BFD *bfd` — указатель на драйвер факторизации базисной матрицы (программный объект типа BFD). Реализация этого драйвера предполагается приватной в том смысле, что подпрограммы симплекс-метода не имеют доступа к полям соответствующего структурного типа.

П.1.2 spx_factorize — вычисление факторизации базисной матрицы

```
int spx_factorize(SPXLP *lp);
```

Данная подпрограмма вычисляет начальную факторизацию текущей базисной матрицы B (1.6).

Если факторизация успешно вычислена, подпрограмма устанавливает признак, что текущая факторизация является действительной, и возвращает нуль. В противном случае подпрограмма сбрасывает указанный признак и возвращает ненулевой код, который был передан драйвером факторизации (`bfd_factorize`).

П.1.3 spx_eval_beta — вычисление значений базисных переменных

```
void spx_eval_beta(SPXLP *lp, double beta[/*1+m*/]);
```

Данная подпрограмма вычисляет вектор $\beta = (\beta_i)$ текущих значений базисных переменных $x_B = [(x_B)_i]$ (2.12). (Факторизация текущей базисной матрицы должна быть действительной.)

Вначале подпрограмма вычисляет модифицированный вектор правых частей ограничений:

$$b' = b - Nf = b - \sum_{j=1}^{n-m} N_j f_j,$$

где $b = (b_i)$ — исходный вектор правых частей ограничений, N — матрица, составленная из столбцов исходной матрицы коэффициентов ограничений A, соответствующих небазис-

ным переменным $f = (f_j)$ — вектор активных границ небазисных переменных $x_N = [(x_N)_j]$, $N_j = A_k$ — столбец матрицы A, соответствующий небазисной переменной $(x_N)_j = x_k$, f_j — значение активной границы $(l_N)_j = l_k$ или $(u_N)_j = u_k$ небазисной переменной $(x_N)_j = x_k$ в текущем базисе. Матрично-векторное произведение Nf вычисляется как линейная комбинация столбцов матрицы N, что позволяет пропускать те столбцы N_j, для которых $f_j = 0$.

Затем подпрограмма выполняет операцию FTRAN для вычисления вектора значений базисных переменных в текущем базисе:

$$\beta = B^{-1}b'.$$

На выходе подпрограмма записывает компоненты вычисленного вектора $\beta = (\beta_i)$ в ячейки массива `beta[1]`, ..., `beta[m]`.

П.1.4 spx_eval_obj — вычисление значения целевой функции

```
double spx_eval_obj(SPXLP *lp, const double beta[/*1+m*/]);
```

Данная подпрограмма вычисляет значение целевой функции в текущем базисном решении (1.13):

$$z = c_B^T\beta + c_N^T f + c_0 = \sum_{i=1}^{m}(c_B)_i\beta_i + \sum_{j=1}^{n-m}(c_N)_j f_j + c_0,$$

где $c_B = [(c_B)_i]$ — вектор коэффициентов целевой функции при базисных переменных, $\beta = (\beta_i)$ — вектор текущих значений базисных переменных, $c_N = [(c_N)_j]$ — вектор коэффициентов целевой функции при небазисных переменных, $f = (f_j)$ — вектор текущих активных границ небазисных переменных, c_0 — постоянный член целевой функции.

Компоненты вектора $\beta = (\beta_i)$ должны находиться в ячейках массива `beta[1]`, ..., `beta[m]`.

П.1.5 spx_eval_pi — вычисление симплексных множителей

```
void spx_eval_pi(SPXLP *lp, double pi[/*1+m*/]);
```

Данная подпрограмма вычисляет вектор симплексных множителей $\pi = (\pi_i)$ (2.13) для текущего базиса. (Факторизация текущей базисной матрицы должна быть действительной.)

Для вычисления вектора π используется операция BTRAN:

$$\pi = B^{-T}c_B,$$

где $c_B = [(c_B)_i]$ — вектор коэффициентов целевой функции при базисных переменных $x_B = [(x_B)_i]$.

На выходе подпрограмма записывает компоненты вычисленного вектора $\pi = (\pi_i)$ в ячейки массива `pi[1]`, ..., `pi[m]`.

П.1.6 spx_eval_dj — вычисление относительной оценки небазисной переменной

```
double spx_eval_dj(SPXLP *lp, const double pi[/*1+m*/], int j);
```

Данная подпрограмма вычисляет относительную оценку d_j небазисной переменной $(x_N)_j = x_k$, $1 \le j \le n - m$, в текущем базисном решении по формуле (2.15):

$$d_j = c_k - A_k^T \pi,$$

где c_k — коэффициент целевой функции при x_k, A_k — k-й столбец матрицы коэффициентов ограничений, π — вектор симплексных множителей для текущего базиса.

Компоненты вектора $\pi = (\pi_i)$ должны находиться в ячейках массива `pi[1]`, ..., `pi[m]`.

П.1.7 spx_eval_tcol — вычисление столбца симплекс-таблицы

```
void spx_eval_tcol(SPXLP *lp, int j, double tcol[/*1+m*/]);
```

Данная подпрограмма вычисляет j-й столбец текущей симплекс-таблицы $\Xi_j = (\xi_{ij})$, $1 \le j \le n - m$. (Факторизация текущей базисной матрицы должна быть действительной.)

Столбец симплекс-таблицы вычисляется по формуле (2.16) с использованием операции FTRAN:

$$\Xi_j = -B^{-1} N_j,$$

где B — текущая базисная матрица, $N_j = A_k$ — столбец матрицы коэффициентов ограничений, который соответствует небазисной переменной $(x_N)_j = x_k$.

На выходе подпрограмма записывает элементы вычисленного столбца $\Xi_j = (\xi_{ij})$ в ячейки массива `tcol[1]`, ..., `tcol[m]`.

П.1.8 spx_eval_rho — вычисление строки обратной базисной матрицы

```
void spx_eval_rho(SPXLP *lp, int i, double rho[/*1+m*/]);
```

Данная подпрограмма вычисляет i-ю строку матрицы B^{-1}, где B — текущая базисная матрица, $1 \le i \le m$. (Факторизация текущей базисной матрицы должна быть действительной.)

Строка обратной матрицы вычисляется по формуле (2.29) с использованием операции BTRAN:

$$\rho = B^{-T} e_i,$$

где e_i — i-й столбец единичной матрицы.

На выходе подпрограмма записывает элементы вычисленной строки $\rho = (\rho_j)$ в ячейки массива `rho[1]`, ..., `rho[m]`.

Примечание. Вектор ρ является вектором симплексных множителей π для целевой функции $z = (x_B)_i$.

П.1.9 spx_eval_tij — вычисление элемента симплекс-таблицы

```
double spx_eval_tij(SPXLP *lp, const double rho[/*1+m*/], int j);
```

Данная подпрограмма вычисляет элемент ξ_{ij} текущей симплекс-таблицы $\Xi = (\xi_{ij}) = -B^{-1}N$, $1 \le i \le m$, $1 \le j \le n - m$, по формуле:

$$\xi_{ij} = e_i^T \Xi e_j = -e_i^T B^{-1} N e_j = -(Ne_j)^T (B^{-T} e_i) = -N_j^T \rho,$$

где e_i и e_j — i-й и j-й столбцы единичной матрицы, $N_j = A_k$ — столбец матрицы коэффициентов ограничний, соответствующий небазисной переменной $(x_N)_j = x_k$, $\rho = B^{-T} e_i$ — i-я строка обратной базисной матрицы B^{-1}.

Компоненты вектора $\rho = (\rho_j)$ должны находиться в ячейках массива `rho[1]`, ..., `rho[m]`.

Примечание. Элемент ξ_{ij} является относительной оценкой d_j небазисной переменной $(x_N)_j$ для целевой функции $z = (x_B)_i$.

П.1.10 spx_eval_trow — вычисление строки симплекс-таблицы

```
void spx_eval_trow(SPXLP *lp, const double rho[/*1+m*/], double
   trow[/*1+n-m*/]);
```

Данная подпрограмма вычисляет i-ю строку текущей симплекс-таблицы $\xi_i = (\xi_{ij})$, $1 \le i \le m$.

Элементы строки симплекс-таблицы $\xi_i = (\xi_{ij})$ для $j = 1, \ldots, n-m$ вычисляются с помощью подпрограммы `spx_eval_tij` (см. выше).

Вектор $\rho = (\rho_j)$, который представляет i-ю строку обратной базисной матрицы B^{-1}, должен быть предварительно вычислен с помощью подпрограммы `spx_eval_rho` (см. выше). Предполагается, что компоненты этого вектора находятся в ячейках массива `rho[1]`, ..., `rho[m]`.

На выходе подпрограмма записывает элементы вычисленной строки $\xi_i = (\xi_{ij})$ в ячейки массива `trow[1]`, ..., `trow[n-m]`.

Примечание. Данная подпрограмма предназначена только для тестирования и отладки.

П.1.11 spx_update_beta — пересчет значений базисных переменных

```
void spx_update_beta(SPXLP *lp, double beta[/*1+m*/], int p, int p_flag, int q,
   const double tcol[/*1+m*/]);
```

Данная подпрограмма выполняет пересчет вектора $\beta = (\beta_i)$ значений базисных переменных $x_B = [(x_B)_i]$ (2.12) для смежного базиса (см. подразд. 2.3).

На входе компоненты вектора β для текущего базиса должны находиться в ячейках массива `beta[1]`, ..., `beta[m]`.

Параметр $1 \le p \le m$ задает номер базисной переменной $(x_B)_p$, которая становится небазисной переменной $(\overline{x}_N)_q$ в смежном базисе. Специальный случай $p < 0$ означает, что в смежном базисе небазисная переменная $(x_N)_q$ переходит со своей текущей (нижней или верхней) активной границы на противоположную, при этом состав базисных и небазисных переменных не изменяется.

Если признак `p_flag` установлен, то в смежном базисе активной границей переменной $(x_B)_p$ становится ее верхняя граница. (В этом случае переменная $(x_B)_p$ должна иметь верхнюю границу и не быть фиксированной переменной.)

Параметр $1 \le q \le n - m$ задает номер небазисной переменной $(x_N)_q$, которая в смежном базисе становится базисной переменной $(\overline{x}_B)_p$ (если $1 \le p \le m$) или переходит со своей текущей активной границы на противоположную (если $p < 0$). (В последнем случае переменная $(x_N)_q$ должна иметь обе верхнюю и нижнюю границы и не быть фиксированной переменной.)

Массив `tcol` должен содержать элементы q-го (ведущего) столбца симплекс-таблицы $\Xi_q = (\xi_{iq})$ в ячейках `tcol[1]`, ..., `tcol[m]`. (Этот столбец должен соответствовать текущему базису.)

Для пересчета значений базисных переменных подпрограмма вначале определяет приращение базисной переменной $(x_B)_p$ в смежном базисе (но только если $1 \le p \le m$):

$$\Delta(x_B)_p = \begin{cases} 0 - \beta_p, & \text{если } -\infty < (x_B)_p < +\infty \\ (l_B)_p - \beta_p, & \text{если } \texttt{p\_flag} = 0 \\ (u_B)_p - \beta_p, & \text{если } \texttt{p\_flag} = 1 \end{cases}$$

где β_p — значение переменной $(x_B)_p$ в текущем базисе, $(l_B)_p$ и $(u_B)_p$ — ее нижняя и верхняя границы. Затем подпрограмма определяет соответствующее приращение небазисной переменной $(x_N)_q$ в смежном базисе:

$$\Delta(x_N)_q = \begin{cases} \Delta(x_B)_p / \xi_{pq}, & \text{если } 1 \le p \le m \\ (u_N)_q - (l_N)_q, & \text{если } p < 0 \text{ и } f_q = (l_N)_q \\ (l_N)_q - (u_N)_q, & \text{если } p < 0 \text{ и } f_q = (u_N)_q \end{cases}$$

где ξ_{pq} — ведущий элемент симплекс-таблицы, f_q — значение активной границы переменной $(x_N)_q$ в текущем базисе.

Если $1 \le p \le m$, то в смежном базисе $(\overline{x}_B)_p = (x_N)_q$, поэтому

$$\overline{\beta}_p = f_q + \Delta(x_N)_q.$$

Значения остальных базисных переменных $(x_B)_i$ для $1 \le i \le m$, $i \ne p$, пересчитываются по формуле:

$$\overline{\beta}_i = \beta_i + \xi_{iq} \Delta(x_N)_q.$$

На выходе подпрограмма записывает компоненты вычисленного вектора $\overline{\beta} = (\overline{\beta}_i)$ в ячейки массива `beta` на место исходного вектора β.

П.1.12 spx_update_d — пересчет относительных оценок небазисных переменных

```
double spx_update_d(SPXLP *lp, double d[/*1+n-m*/], int p, int q, const double
   trow[/*1+n-m*/], const double tcol[/*1+m*/]);
```

Данная подпрограмма выполняет пересчет вектора $d = (d_j)$ относительных оценок небазисных переменных $x_N = [(x_N)_j]$ (2.14) для смежного базиса (см. подразд. 2.3).

На входе компоненты вектора d для текущего базиса должны находиться в ячейках массива `d[1]`, ..., `d[n-m]`.

Параметр $1 \le p \le m$ задает номер базисной переменной $(x_B)_p$, которая становится небазисной переменной $(\overline{x}_N)_q$ в смежном базисе.

Параметр $1 \le q \le n - m$ задает номер небазисной переменной $(x_N)_q$, которая становится базисной переменной $(\overline{x}_B)_p$ в смежном базисе.

Массив `trow` должен содержать элементы p-й (ведущей) строки симплекс-таблицы $\xi_p = (\xi_{pj})$ в ячейках `trow[1]`, ..., `trow[n-m]`. (Эта строка должна соответствовать текущему базису.)

Массив `tcol` должен содержать элементы q-го (ведущего) столбца симплекс-таблицы $\Xi_q = (\xi_{iq})$ в ячейках `tcol[1]`, ..., `tcol[m]`. (Этот столбец должен соответствовать текущему базису.)

Для пересчета относительных оценок подпрограмма вначале вычисляет более точное значение оценки d_q для текущего базиса по формуле (2.27), используя q-й столбец симплекс-таблицы Ξ_q:

$$d_q = (c_N)_q + \sum_{i=1}^{m} \xi_{iq}(c_B)_i,$$

где $(c_N)_q$ и $(c_B)_i$ — коэффициенты целевой функции при переменных $(x_N)_q$ и $(x_B)_i$, соответственно. Также подпрограмма вычисляет относительную ошибку:

$$e = \frac{|d_q - d'_q|}{1 + |d_q|},$$

где d'_q — менее точное значение оценки d_q на входе в подпрограмму, и возвращает значение e в качестве результата. (Вызывающая программа может выполнить прямое вычисление относительных оценок, если величина e оказывается слишком большой, так как в этом случае другие относительные оценки вероятно тоже являются недостаточно точными.)

Затем подпрограмма вычисляет относительную оценку переменной $(\overline{x}_N)_q = (x_B)_p$ в смежном базисе по формуле (2.25):

$$\overline{d}_q = \frac{d_q}{\xi_{pq}},$$

где ξ_{pq} — ведущий элемент симплекс-таблицы (его значение берется из ведущего столбца Ξ_q как более точное), а также относительные оценки остальных небазисных переменных $(x_N)_j$ для $1 \le j \le n - m$, $j \ne q$, по формуле (2.26):

$$\overline{d}_j = d_j - \xi_{pj}\overline{d}_q,$$

где ξ_{pj} — элемент p-й строки симплекс-таблицы ξ_p.

На выходе подпрограмма записывает компоненты вычисленного вектора $\overline{d} = (\overline{d}_j)$ в ячейки массива `d` на место исходного вектора d.

П.1.13 spx_change_basis — переход к смежному базису

```
void spx_change_basis(SPXLP *lp, int p, int p_flag, int q);
```

Данная подпрограмма выполняет переход к смежному базису, изменяя заголовок базиса (поля `lp->head` и `lp->flag`).

Параметры `p`, `p_flag` и `q` имеют тот же смысл, что и для подпрограммы `spx_update_beta`.

П.1.14 spx_update_invb — пересчет факторизации базисной матрицы

```
int spx_update_invb(SPXLP *lp, int i, int k);
```

Данная подпрограмма выполняет пересчет факторизации базисной матрицы B (1.6) после замены ее p-го столбца k-м столбцом матрицы коэффициентов ограничений A (1.2).

Параметр $1 \leq p \leq m$ задает номер столбца матрицы B, который должен быть заменен.

Параметр $1 \leq k \leq n$ задает номер столбца матрицы A, который используется для замены.

Если пересчет факторизации успешно выполнен, подпрограмма устанавливает признак, что текущая факторизация является действительной, и возвращает нуль. В противном случае подпрограмма сбрасывает указанный признак и возвращает ненулевой код, который был передан драйвером факторизации (`bfd_update`).

П.2 Модуль SPXAT — матрица A в строчном формате

П.2.1 Структурный тип SPXAT

Структурный тип данных `SPXAT` используется для представления матрицы коэффициентов ограничений A, имеющей m строк и n столбцов, в разреженном строчном формате.

Данная структура логически связана со структурой `SPXLP` и имеет следующие поля:

`int *ptr` — указатель на массив длины $1 + m + 1$. Ячейка `ptr[0]` не используется. Ячейка `ptr[i]`, $1 \le i \le m$, определяет начальную позицию i-й строки матрицы A в массивах `ind` и `val`, при этом всегда `ptr[1]` $= 1$ (т. е. первая строка начинается в первой позиции). Величина `ptr[m+1]` $= nnz+1$ определяет позицию, следующую за последним элементом в массивах `ind` и `val` (т. е. позицию, в которой бы начиналась $(m+1)$-я строка). Строки матрицы следуют друг за другом без промежутков, поэтому длина (число ненулевых элементов) i-й строки матрицы равна `ptr[i+1]` $-$ `ptr[i]` ≥ 0.

`int *ind` — указатель на массив длины $1 + nnz$. Ячейка `ind[0]` не используется. Ячейки `ind[1]`, ..., `ind[nnz]` содержат столбцовые индексы соответствующих элементов матрицы A.

`double *val` — указатель на массив длины $1+nnz$. Ячейка `val[0]` не используется. Ячейки `val[1]`, ..., `val[nnz]` содержат (ненулевые) значения соответствующих элементов матрицы A.

`double *work` — указатель на рабочий массив длины $1 + n$.

П.2.2 spx_alloc_at — размещение массивов

```
void spx_alloc_at(SPXLP *lp, SPXAT *at);
```

Данная подпрограмма выполняет отведение памяти для всех массивов, входящих в состав объекта `at` (матрица коэффициентов ограничений A в разреженном строчном формате), используя для этого информацию о размерности рабочей ЛП-задачи.

П.2.3 spx_build_at — формирование матрицы A в строчном формате

```
void spx_build_at(SPXLP *lp, SPXAT *at);
```

Используя представление матрицы коэффициентов ограничений A в разреженном столбцовом формате, которое содержится в объекте `lp`, данная подпрограмма формирует представление этой матрицы в разреженном строчном формате и записывает результат в объект `at`.

Вызову данной подпрограммы должен предшествовать вызов подпрограммы `spx_alloc_at`.

П.2.4 spx_at_prod — вычисление матричного произведения $\overline{y} = y + sA^Tx$

```
void spx_at_prod(SPXLP *lp, SPXAT *at, double y[/*1+n*/], double s, const double
   x[/*1+m*/]);
```

Данная подпрограмма предназначена для вычисления матричного произведения:

$$\overline{y} = y + sA^Tx,$$

где A — $m \times n$-матрица коэффициентов ограничений, x — заданный m-вектор, s — заданный скаляр, y — заданный n-вектор.

На входе в попдрограмму элементы векторов x и y должны находиться в ячейках массивов `x[1]`, ..., `x[m]` и `y[1]`, ..., `y[n]`, соответственно. На выходе подпрограмма записывает элементы вычисленного вектора $\overline{y}$ в массив `y` на место исходного вектора y.

Подпрограмма использует строчное представление матрицы A и вычисляет результирующий вектор как линейную комбинацию строк этой матрицы:

$$\overline{y} = y + s \sum_{i=1}^{m} a_i x_i,$$

где a_i — i-я строка матрицы A.

П.2.5 spx_nt_prod1 — вычисление матричного произведения $\overline{y} = y + sN^T x$

```
void spx_nt_prod1(SPXLP *lp, SPXAT *at, double y[/*1+n-m*/], int ign, double s,
   const double x[/*1+m*/]);
```

Данная подпрограмма предназначена для вычисления матричного произведения:

$$\overline{y} = y + sN^T x,$$

где N — $m \times (n-m)$-матрица (1.6), составленная из столбцов матрицы коэффициентов ограничений A, соответствующих небазисным переменным x_N, x — заданный m-вектор, s — заданный скаляр, y — заданный $(n-m)$-вектор.

На входе в попдрограмму элементы векторов x и y должны находиться в ячейках массивов `x[1]`, ..., `x[m]` и `y[1]`, ..., `y[n-m]`, соответственно. Если установлен признак `ign`, то исходное содержимое массива `y` игнорируется и считается, что $y = 0$. На выходе подпрограмма записывает элементы вычисленного вектора $\overline{y}$ на место исходного вектора y.

Из равенства (1.6) следует, что:

$$A^T = \Pi^T \begin{pmatrix} B^T \\ N^T \end{pmatrix}.$$

Поэтому:

$$\begin{pmatrix} \cdot \\ \overline{y} \end{pmatrix} = \begin{pmatrix} 0 + sB^T x \\ y + sN^T x \end{pmatrix} = \Pi \left[\Pi^T \begin{pmatrix} 0 \\ y \end{pmatrix} + s\Pi^T \begin{pmatrix} B^T \\ N^T \end{pmatrix} x \right] =$$

$$= \Pi \left[\Pi^T \begin{pmatrix} 0 \\ y \end{pmatrix} + sA^T x \right].$$

В соответствии с этой формулой подпрограмма вначале вычисляет вспомогательный n-вектор:

$$w = \Pi^T \begin{pmatrix} y \\ 0 \end{pmatrix},$$

устанавливая $w_k = y_j$, если $x_k = (x_N)_j$, далее вычисляет вектор

$$\overline{w} = w + sA^T x,$$

используя подпрограмму `spx_at_prod`, после чего формирует результирующий вектор $\overline{y}$, используя соответствующие элементы вектора $\overline{w}$:

$$\begin{pmatrix} \cdot \\ \overline{y} \end{pmatrix} = \Pi \overline{w}.$$

П.2.6 spx_eval_trow1 — вычисление строки симплекс-таблицы

```
void spx_eval_trow1(SPXLP *lp, SPXAT *at, const double rho[/*1+m*/], double
   trow[/*1+n-m*/]);
```

Данная подпрограмма вычисляет i-ю строку текущей симплекс-таблицы $\xi_i = (\xi_{ij})$, $1 \le i \le m$, используя строчное представление матрицы коэффициентов ограничений A.

Вектор $\rho = (\rho_j)$, который представляет i-ю строку обратной базисной матрицы B^{-1}, должен быть предварительно вычислен с помощью подпрограммы `spx_eval_rho`. Предполагается, что компоненты этого вектора находятся в ячейках массива `rho[1]`, ..., `rho[m]`.

Имеется два способа вычисления строки симплекс-таблицы.

1. Элементы ξ_{ij} этой строки вычисляются как скалярные произведения для $j = 1, \ldots, n-m$:

$$\xi_{ij} = -N_j^T \rho = -A_k^T \rho = -\sum_{i=1}^{m} a_{ik}\rho_i,$$

где $N_j = A_k$ — столбец матрицы коэффициентов ограничений, соответствующий небазисной переменной $(x_N)_k$. Число операций в данном случае равно числу ненулевых элементов матрицы N, так что оценка этого числа равна:

$$\nu_1 = (n-m)\frac{nnz(A)}{n},$$

где $(n-m)$ — число столбцов матрицы N, $nnz(A)/n$ — среднее число ненулевых элементов в отдельном столбце матрицы A, а значит, и матрицы N.

2. Строка ξ_i вычисляется как часть линейной комбинации строк матрицы A, взятых с коэффициентами $\rho_i \ne 0$. Оценка числа операций для этого способа равна:

$$\nu_2 = nnz(\rho)\frac{nnz(A)}{m},$$

где $nnz(\rho)$ — число ненулевых элементов вектора ρ, $nnz(A)/m$ — среднее число ненулевых элементов в отдельной строке матрицы A.

Если $\nu 1 < \nu 2$, подпрограмма применяет первый способ (аналогично подпрограмме `spx_eval_trow`, но не вызывая внешних подпрограмм). Если же $\nu 1 \ge \nu 2$, подпрограмма применяет второй способ, используя для этого подпрограмму `spx_nt_prod1`.

На выходе подпрограмма записывает элементы вычисленной строки $\xi_i = (\xi_{ij})$ в ячейки массива `trow[1]`, ..., `trow[n-m]`.

П.2.7 spx_free_at — освобождение массивов

```
void spx_free_at(SPXLP *lp, SPXAT *at);
```

Данная подпрограмма выполняет освобождение памяти, занятой всеми массивами, входящими в состав объекта `at`.

П.3 Модуль SPXNT — матрица N в строчном формате

П.3.1 Структурный тип SPXNT

Структурный тип данных `SPXNT` используется для представления матрицы N, состоящей из небазисных столбцов матрицы коэффициентов ограничений A и имеющей m строк и $n - m$ столбцов, в разреженном строчном формате.

Данная структура логически связана со структурой `SPXLP` и имеет следующие поля:

`int *ptr` — указатель на массив длины $1 + m$. Ячейка `ptr[0]` не используется. Ячейка `ptr[i]`, $1 \leq i \leq m$, определяет начальную позицию i-й строки матрицы N в массивах `ind` и `val`, при этом всегда `ptr[1]` $= 1$ (т. е. первая строка начинается в первой позиции). Эти указатели строк формируются один раз как если бы они соответствовали строкам матрицы A, расположенным без промежутков, т. е. `ptr[i+1]` – `ptr[i]` есть длина (число ненулевых элементов) i-й строки ($i < m$) матрицы A и ($nnz + 1$) – `ptr[m]` есть длина m-й (последней) строки матрицы A, где nnz — общее число ненулевых элементов в матрице A.

`int *len` — указатель на массив длины $1 + m$. Ячейка `len[0]` не используется. Ячейка `len[i]`, $1 \leq i \leq m$, определяет текущую длину (число ненулевых элементов) i-й строки матрицы N.

`int *ind` — указатель на массив длины $1 + nnz$. Ячейка `ind[0]` не используется. Ячейки `ind[1]`, ..., `ind[nnz]` содержат столбцовые индексы соответствующих элементов матрицы N.

`double *val` — указатель на массив длины $1+nnz$. Ячейка `val[0]` не используется. Ячейки `val[1]`, ..., `val[nnz]` содержат (ненулевые) значения соответствующих элементов матрицы N.

П.3.2 spx_alloc_nt — размещение массивов

```
void spx_alloc_nt(SPXLP *lp, SPXNT *nt);
```

Данная подпрограмма выполняет отведение памяти для всех массивов, входящих в состав объекта `nt` (матрица N, состоящая из небазисных столбцов матрицы коэффициентов ограничений A и хранимая в разреженном строчном формате), используя для этого информацию о размерности рабочей ЛП-задачи.

П.3.3 spx_init_nt — инициализация указателей строк

```
void spx_init_nt(SPXLP *lp, SPXNT *nt);
```

Данная подпрограмма выполняет инициализацию (начальную установку) указателей строк матрицы N, используя столбцовое представление матрицы коэффициентов ограничений A.

Данную подпрограмму достаточно вызвать один раз непосредственно после вызова подпрограммы `spx_alloc_nt`.

П.3.4 spx_nt_add_col — добавление элементов столбца

```
void spx_nt_add_col(SPXLP *lp, SPXNT *nt, int j, int k);
```

Данная подпрограмма добавляет элементы столбца $N_j = A_k$, где $1 \le j \le n - m$, $1 \le k \le n$, в строчное представление матрицы N. Без проверки предполагается, что элементы этого столбца отсутствуют в строчном представлении N.

П.3.5 spx_build_nt — формирование матрицы N для текущего базиса

```
void spx_build_nt(SPXLP *lp, SPXNT *nt);
```

Данная подпрограмма формирует строчное представление матрицы N для текущего базиса, добавляя в эту матрицу столбцы матрицы коэффициентов ограничений A, соответствующие небазисным переменным.

П.3.6 spx_nt_del_col — удаление элементов столбца

```
void spx_nt_del_col(SPXLP *lp, SPXNT *nt, int j, int k);
```

Данная подпрограмма удаляет элементы столбца $N_j = A_k$, где $1 \le j \le n - m$, $1 \le k \le n$, из строчного представления матрицы N. Без проверки предполагается, что элементы указанного столбца присутствуют в строчном представлении N.

П.3.7 spx_update_nt — обновление матрицы N для смежного базиса

```
void spx_update_nt(SPXLP *lp, SPXNT *nt, int p, int q);
```

Данная подпрограмма обновляет строчное представление матрицы N для смежного базиса, где q-й столбец этой матрицы, $1 \le q \le n - m$, заменяется p-м столбцом текущей базисной матрицы B, $1 \le p \le m$.

П.3.8 spx_nt_prod — вычисление матричного произведения $\overline{y} = y + sN^T x$

```
void spx_nt_prod(SPXLP *lp, SPXNT *nt, double y[/*1+n-m*/], int ign, double s,
   const double x[/*1+m*/]);
```

Данная подпрограмма предназначена для вычисления матричного произведения:

$$\overline{y} = y + sN^T x,$$

где N — $m \times (n-m)$-матрица (1.6), составленная из столбцов матрицы коэффициентов ограничений A, соответствующих небазисным переменным x_N, x — заданный m-вектор, s — заданный скаляр, y — заданный $(n - m)$-вектор.

На входе в попдрограмму элементы векторов x и y должны находиться в ячейках массивов `x[1]`, ..., `x[m]` и `y[1]`, ..., `y[n-m]`, соответственно. Если установлен признак `ign`, то исходное содержимое массива `y` игнорируется и считается, что $y = 0$. На выходе подпрограмма записывает элементы вычисленного вектора $\overline{y}$ на место исходного вектора y.

Подпрограмма использует строчное представление матрицы N и вычисляет результирующий вектор как линейную комбинацию строк этой матрицы:

$$\overline{y} = y + s \sum_{i=1}^{m} \nu_i x_i,$$

где ν_i — i-я строка матрицы N.

П.3.9 spx_free_nt — освобождение массивов

```
void spx_free_nt(SPXLP *lp, SPXNT *nt);
```

Данная подпрограмма выполняет освобождение памяти, занятой всеми массивами, входящими в состав объекта `nt`.

П.4 Модуль SPXCHUZC — выбор небазисной переменной

П.4.1 spx_chuzc_sel — отбор подходящих переменных

```
int spx_chuzc_sel(SPXLP *lp, const double d[/*1+n-m*/], double tol, double tol1,
   int list[/*1+n-m*/]);
```

Данная подпрограмма выполняет отбор подходящих небазисных переменных, т. е. небазисных переменных $(x_N)_j$, относительная оценка d_j которых имеет «неправильный» знак, вследствие чего изменение такой переменной в допустимом направлении приводит к улучшению (уменьшению) целевой функции.

Компоненты вектора относительных оценок небазисных переменных $d = (d_j)$ должны находится в ячейках массива `d[1]`, ..., `d[n-m]`.

Небазисная переменная $(x_N)_j$ считается подходящей, если:

$$d_j \le -\varepsilon_j \text{ и } (x_N)_j \text{ может возрастать}$$
$$d_j \ge +\varepsilon_j \text{ и } (x_N)_j \text{ может убывать}$$

для

$$\varepsilon_j = tol + tol_1|(c_N)_j|,$$

где $(c_N)_j$ — коэффициент целевой функции при переменной $(x_N)_j$, tol и tol_1 — заданные допуски.

Примечание. Так как увеличение коэффициентов целевой функции в k раз увеличивает относительные оценки также в k раз, то заданные допуски tol и tol_1 должны быть соответствующим образом промасштабированы.

На выходе подпрограмма записывает номера j подходящих небазисных переменных $(x_N)_j$ в ячейки массива `list[1]`, ..., `list[num]`, где общее число таких переменных $0 \le$ `num` $\le n-m$ возвращается в качестве результата. (Если указатель на массив `list` задан как `NULL`, то номера небазисных переменных не записываются.)

П.4.2 spx_chuzc_std — выбор небазисной переменной по правилу Данцига

```
int spx_chuzc_std(SPXLP *lp, const double d[/*1+n-m*/], int num, const int
   list[]);
```

Данная подпрограмма выполняет «стандартный» выбор наиболее подходящей небазисной переменной $(x_N)_q$ в соответствии с правилом Данцига (см. подразд. 3.1):

$$|d_q| = \max_{j \in J} |d_j|,$$

где $J \subseteq \{1, \ldots, n-m\}$ — множество номеров подходящих небазисных переменных, d_j — относительная оценка небазисной переменной $(x_N)_j$ в текущем базисе.

Компоненты вектора относительных оценок небазисных переменных $d = (d_j)$ должны находится в ячейках массива `d[1]`, ..., `d[n-m]`.

Номера подходящих небазисных переменных $j \in J$ должны находится в ячейках массива `list[1]`, ..., `list[num]`, где `num` $= |J| > 0$ — общее число таких переменных.

На выходе подпрограмма возвращает номер q выбранной небазисной переменной $(x_N)_q$.

П.4.3 Структурный тип SPXSE

Структурный тип данных `SPXSE` используется подпрограммами, реализующими проекционный метод наиболее крутого ребра (см. подразд. 3.3) и метод оценивания Devex (см. подразд. 3.4).

Структура `SPXSE` имеет следующие поля.

`int valid` — если данный признак установлен, то содержимое структуры является действительным. В противном случае, если данный признак сброшен, содержимое структуры не действительно.

`char *refsp` — указатель на массив длины $1+n$. Ячейка `refsp[0]` не используется. Ячейка `refsp[k]`, $1 \le k \le n$, содержит признак того, что переменная x_k принадлежит эталонному пространству переменных $V \subseteq E^n$.

`double *gamma` — указатель на массив длины $1 + n - m$. Ячейка `gamma[0]` не используется. Ячейка `gamma[j]`, $1 \le j \le n - m$, содержит весовой множитель γ_j (3.27) для относительной оценки d_j небазисной переменной $(x_N)_j$ в текущем базисе.

`double *work` — указатель на рабочий массив длины $1 + m$.

П.4.4 spx_alloc_se — размещение массивов

```
void spx_alloc_se(SPXLP *lp, SPXSE *se);
```

Данная подпрограмма выполняет отведение памяти для всех массивов, входящих в состав объекта `se` (блок данных проекционного метода наиболее крутого ребра и метода оценивания Devex), используя для этого информацию о размерности рабочей ЛП-задачи.

П.4.5 spx_reset_refsp — переопределение эталонного пространства

```
void spx_reset_refsp(SPXLP *lp, SPXSE *se);
```

Данная подпрограмма выполняет переопределение эталонного пространства, включая в него переменные, которые являются небазисными в текущем базисе, а также устанавливает все весовые множители γ_j равными единице.

П.4.6 spx_eval_gamma_j — прямое вычисление весового множителя

```
double spx_eval_gamma_j(SPXLP *lp, SPXSE *se, int j);
```

Данная подпрограмма выполняет непосредственное вычисление весового множителя γ_j, $1 \le j \le n - m$, для текущего базиса по формуле (3.27).

П.4.7 spx_chuzc_pse — выбор небазисной переменной по методу наиболее крутого ребра

```
int spx_chuzc_pse(SPXLP *lp, SPXSE *se, const double d[/*1+n-m*/], int num,
   const int list[]);
```

Данная подпрограмма выполняет выбор наиболее подходящей небазисной переменной $(x_N)_q$ в соответствии с проекционным методом наиболее крутого ребра (3.32):

$$\frac{d_q^2}{\gamma_q} = \max_{j \in J} \frac{d_j^2}{\gamma_j},$$

где $J \subseteq \{1, \ldots, n-m\}$ — множество номеров подходящих небазисных переменных, d_j — относительная оценка небазисной переменной $(x_N)_j$ в текущем базисе, γ_j — весовой множитель относительной оценки.

Компоненты вектора относительных оценок небазисных переменных $d = (d_j)$ должны находится в ячейках массива `d[1]`, ..., `d[n-m]`.

Номера подходящих небазисных переменных $j \in J$ должны находится в ячейках массива `list[1]`, ..., `list[num]`, где `num` $= |J| > 0$ — общее число таких переменных.

На выходе подпрограмма возвращает номер q выбранной небазисной переменной $(x_N)_q$.

П.4.8 spx_update_gamma — пересчет весовых множителей для смежного базиса

```
double spx_update_gamma(SPXLP *lp, SPXSE *se, int p, int q, const double
   trow[/*1+n-m*/], const double tcol[/*1+m*/]);
```

Данная подпрограмма выполняет точный пересчет вектора весовых множителей $\gamma = (\gamma_j)$ (3.27), используемых в проекционном методе наиболее крутого ребра, для смежного базиса.

Содержимое объекта `se` должно быть действительным и соответствовать текущему базису.

Параметр $1 \le p \le m$ задает номер базисной переменной $(x_B)_p$, которая становится небазисной переменной $(\overline{x}_N)_q$ в смежном базисе.

Параметр $1 \le q \le n-m$ задает номер небазисной переменной $(x_N)_q$, которая становится базисной переменной $(\overline{x}_B)_p$ в смежном базисе.

Массив `trow` должен содержать элементы p-й (ведущей) строки симплекс-таблицы $\xi_p = (\xi_{pj})$ в ячейках `trow[1]`, ..., `trow[n-m]`. (Эта строка должна соответствовать текущему базису.)

Массив `tcol` должен содержать элементы q-го (ведущего) столбца симплекс-таблицы $\Xi_q = (\xi_{iq})$ в ячейках `tcol[1]`, ..., `tcol[m]`. (Этот столбец должен соответствовать текущему базису.)

Для пересчета весовых множителей подпрограмма использует алгоритм, приведенный в подразд. 3.3.

Подпрограмма также вычисляет относительную ошибку:

$$e = \frac{|\gamma_q - \gamma'_q|}{1 + |\gamma_q|},$$

где γ'_q — менее точное значение весового множителя γ_q на входе в подпрограмму, и возвращает e в качестве результата. (Вызывающая программа может переопределить эталонное пространство, если величина e оказывается слишком большой, так как в этом случае другие весовые множители вероятно тоже являются недостаточно точными.)

П.4.9 spx_free_se — освобождение массивов

```
void spx_free_se(SPXLP *lp, SPXSE *se);
```

Данная подпрограмма выполняет освобождение памяти, занятой всеми массивами, входящими в состав объекта `se`.

П.5 Модуль SPXCHUZR — выбор базисной переменной

П.5.1 spx_chuzr_std — «стандартный» выбор базисной переменной

```
int spx_chuzr_std(SPXLP *lp, int phase, const double beta[/*1+m*/],
   int q, double s, const double tcol[/*1+m*/], int *p_flag, double tol_piv,
   double tol, double tol1);
```

Данная подпрограмма реализует улучшенный «стандартный» метод выбора базисной переменной $(x_B)_p$ (см. подразд. 4.2) применительно к ЛП-задаче в рабочем формате (1.1)—(1.3).

Параметр `phase` задает этап поиска:

1 — поиск допустимого решения. В этом случае подпрограмма использует фиктивные границы базисных переменных, соответствующие точкам излома штрафной функции:

$$(\widetilde{l}_B)_i = \begin{cases} (l_B)_i, & \text{если } (c_B)_i = 0 \\ (u_B)_i, & \text{если } (c_B)_i > 0 \\ -\infty, & \text{если } (c_B)_i < 0 \end{cases}$$

$$(\widetilde{u}_B)_i = \begin{cases} (u_B)_i, & \text{если } (c_B)_i = 0 \\ +\infty, & \text{если } (c_B)_i > 0 \\ (l_B)_i, & \text{если } (c_B)_i < 0 \end{cases}$$

где $(l_B)_i$ и $(u_B)_i$ — исходные (оригинальные) границы переменной $(x_B)_i$, $(c_B)_i$ — коэффициент штрафной функции при этой переменной.

2 — поиск оптимального решения. В этом случае подпрограмма использует исходные (оригинальные) границы базисных переменных.

Компоненты вектора значений базисных переменных $\beta = (\beta_i)$ должны находиться в ячейках массива `beta[1]`, ..., `beta[m]`. Предполагается, что в пределах допуска эти значения находятся внутри границ соответствующих базисных переменных.

Параметр $1 \le q \le n - m$ задает номер выбранной небазисной переменной $(x_N)_q$.

Параметр s задает направление изменения выбранной небазисной переменной $(x_N)_q$: если $s = +1.0$, то $(x_N)_q$ возрастает, а если $s = -1.0$, то $(x_N)_q$ убывает. (Таким образом, параметр соответствующего луча есть $\theta = s[(x_N)_q - f_q] \ge 0$, где f_q — значение активной границы $(x_N)_q$ в текущем базисном решении.)

Элементы столбца текущей симплекс-таблицы $\Xi_q = (\xi_{iq})$, соответствующего выбранной небазисной переменной $(x_N)_q$, должны находиться в ячейках массива `tcol[1]`, ..., `tcol[m]`.

Параметр `tol_piv` задает допуск на абсолютную величину элементов симплекс-таблицы. Если $|\xi_{iq}| <$ `tol_piv`, то базисная переменная $(x_B)_i$ пропускается, т. е. считается, что эта переменная не зависит от параметра луча θ.

Параметры `tol` и `tol1` задают допуски, которые используются для создания искусственной вырожденности с целью расширения свободы выбора ведущего элемента. Если $\beta_i \le (l_B)_i + \delta_i$, где $\delta_i = tol + tol_1|(l_B)_i|$, то считается, что в пределах допуска β_i совпадает с нижней границей $(l_B)_i$. Аналогично, если $\beta_i \ge (u_B)_i - \delta_i$, где $\delta_i = tol + tol_1|(u_B)_i|$, то считается, что в пределах допуска β_i совпадает с верхней границей $(u_B)_i$.

Подпрограмма определяет номер $1 \le p \le m$ базисной переменной $(x_B)_p$, которая первой достигает своей (нижней или верхней) границы при возрастании параметра луча $\theta \ge 0$, записывает в ячейку, заданную указателем `p_flag`, признак этой границы (0 — нижняя граница

или фиксированное значение, 1 — верхняя граница), и возвращает номер p в качестве результата. Если же первой достигает своей противоположной границы небазисная переменная $(x_N)_q$, то подпрограмма в качестве результата возвращает (-1). В том случае, когда возможно неограниченное возрастание параметра луча θ, подпрограмма возвращает нуль.

Примечание. Признак границы, записываемый в ячейку `p_flag`, относится к исходной (оригинальной) границе переменной $(x_B)_p$ и определяет признак активной границы `lp->flag[q]`, который должен быть установлен для указанной переменной в смежном базисе.

П.5.2 spx_chuzr_harris — выбор базисной переменной по методу Харрис

```
int spx_chuzr_harris(SPXLP *lp, int phase, const double beta[/*1+m*/],
   int q, double s, const double tcol[/*1+m*/], int *p_flag, double tol_piv,
   double tol, double tol1);
```

Данная подпрограмма реализует метод Харрис выбора базисной переменной $(x_B)_p$ (см. подразд. 4.3) применительно к ЛП-задаче в рабочем формате (1.1)—(1.3).

Все параметры подпрограммы, за исключением `tol` и `tol1`, а также возвращаемое значение имеют тот же смысл, что и для подпрограммы `spx_chuzr_std` (см. выше).

Параметры `tol` и `tol1` задают допуски на нарушение границ базисных переменных. Для нижней границы переменной $(x_B)_i$ этот допуск принимается равным $\delta_i = tol + tol_1|(l_B)_i|$, а для верхней границы этот допуск принимается равным $\delta_i = tol + tol_1|(u_B)_i|$.

П.6 Модуль SPXPROB — интерфейс для пакета GLPK

Программный модуль SPXPROB реализует интерфейс между программами симплекс-метода и пакетом GLPK. Основными функциями интерфейса являются:

1. Преобразование исходной задачи из обобщенного формата GLPK в рабочий формат. Это преобразование может дополнительно включать масштабирование задачи, исключение небазисных фиксированных переменных и смещение границ переменных.

2. Обратное преобразование полученного базисного решения рабочей задачи к базисному решению исходной задачи в обобщенном формате GLPK. Это преобразование включает обратное смещение значений переменных, вычисление относительных оценок исключенных небазисных переменных и демасштабирование всех компонент решения.

П.6.1 Преобразование ЛП-задачи к рабочему формату

На уровне прикладной программы в пакете GLPK используется *обобщенный* формат ЛП-задачи:

$$z = c^T x + c_0 \to \min / \max \tag{П.1}$$

$$L \le Ax \le U \tag{П.2}$$

$$l \le x \le u \tag{П.3}$$

который имеет следующие два основных отличия от рабочего формата (1.1)—(1.3):

возможна как минимизация, так и максимизация целевой функции (рабочий формат предполагает только минимизацию);

ограничения общего вида (П.2) могут иметь те же комбинации границ, что и ограничения на переменные (П.3) (в рабочем формате все ограничения (1.2) имеют вид равенств).

Переход от обобщенного формата (П.1)—(П.3) к рабочему формату (1.1)—(1.3) осуществляется следующим образом.

Рассмотрим i-е ограничение исходной системы ограничений (П.2):

$$L_i \le \sum_j a_{ij}(x_S)_j \le U_i, \tag{П.4}$$

где $(x_S)_j$ — *структурная переменная*, т. е. переменная x_j в обозначениях исходной ЛП-задачи (П.1)—(П.3). Для преобразования к рабочему формату это ограничение заменяется следующим ограничением, имеющим вид равенства:

$$(x_R)_i = \sum_j a_{ij}(x_S)_j, \tag{П.5}$$

где $(x_R)_i$ — *вспомогательная переменная* i-го ограничения. Значение $(x_R)_i$ совпадает со значением соответствующей линейной формы исходного ограничения, что позволяет перенести исходные границы линейной формы на эту вспомогательную переменную:

$$L_i \le (x_R)_i \le U_i. \tag{П.6}$$

Очевидно, что ограничения (П.5) и (П.6) эквивалентны исходному ограничению (П.4).

Примечание. В некоторых пакетах линейного программирования преобразование ограничений-неравенств к виду равенств выполняется иначе. Так, ограничение-неравенство $\sum a_{ij}x_j \le b_i$ заменяется ограничением $\sum a_{ij}x_j + y_i = b_i$, где $y_i \ge 0$ — неотрицательная *переменная недостатка*, а ограничение-неравенство $\sum a_{ij}x_j \ge b_i$ заменяется ограничением $\sum a_{ij}x_j - y_i = b_i$, где $y_i \ge 0$ — неотрицательная *переменная избытка*. Такой способ преобразования в основном объясняется историческими причинами, поскольку в ранних реализациях симплекс-метода применялся *стандартный формат* ЛП-задачи, где вместо границ переменных общего вида (1.3) использовались более простые условия неотрицательности переменных.

В результате включения вспомогательных переменных ЛП-задача (П.1)—(П.3) будет иметь следующий вид:

$$\begin{gathered} z = 0^T x_R + c^T x_S + c_0 \to \min / \max \\ x_R = A x_S \\ L \le x_R \le U \\ l \le x_S \le u \end{gathered} \tag{П.7}$$

где $x_R = [(x_R)_i]$ — вектор вспомогательных переменных, $x_S = [(x_S)_j]$ — вектор структурных (исходных) переменных. (Заметим, что вспомогательные переменные входят в целевую функцию с нулевыми коэффициентами.)

С формальной точки зрения нет никакого различия между структурными и вспомогательными переменными, поэтому все эти переменные можно объединить в один вектор переменных. Это позволяет считать, что ЛП-задача (П.7) представлена в рабочем формате (если не принимать во внимание случай максимизации):

$$z = \begin{pmatrix} 0 \\ c \end{pmatrix}^T \begin{pmatrix} x_R \\ x_S \end{pmatrix} + c_0 \to \min / \max \tag{П.8}$$

$$(I \mid -A) \begin{pmatrix} x_R \\ x_S \end{pmatrix} = 0 \tag{П.9}$$

$$\begin{pmatrix} L \\ l \end{pmatrix} \le \begin{pmatrix} x_R \\ x_S \end{pmatrix} \le \begin{pmatrix} U \\ u \end{pmatrix} \tag{П.10}$$

Важно отметить, что вспомогательные переменные включаются во *все* ограничения (П.2) исходной ЛП-задачи, даже если некоторые из них уже имеют вид равенств (т. е. для которых $L_i = U_i$), так что *расширенная матрица коэффициентов ограничений* $(I \mid -A)$ всегда содержит единичную подматрицу I, соответствующую единичным коэффициентам при вспомогательных переменных. Это гарантирует, что указанная матрица всегда имеет полный строчный ранг, т. е. все ее строки, а значит, и все ограничения-равенства (П.9) являются линейно независимыми, причем даже в тех случаях, когда в исходной ЛП-задаче (П.1)—(П.3) имеются линейно зависимые ограничения-равенства.

П.6.2 Масштабирование ЛП-задачи

Пакет GLPK предусматривает возможность масшатбирования исходных данных ЛП-задачи. Масштабирование обычно применяется для улучшения обусловленности матрицы коэффициентов ограничений, что позволяет повысить точность вычислений, в частности, при выполнении основных операцих с базисной матрицей (см. подразд. 2.1). Заметим, что на уровне

прикладной программы исходные данные хранятся в немасштабированном виде — масштабирование выполняется во время преобразования ЛП-задачи к рабочему формату.

В пакете GLPK предполагается, что матрица коэффициентов ограничений масштабированной ЛП-задачи имеет вид:

$$\widetilde{A} = RAS, \tag{П.11}$$

где A — матрица коэффициентов ограничений исходной (немасштабированной) ЛП-задачи (П.1)—(П.3), $R = \mathrm{diag}(r_{ii})$ — заданная диагональная матрица масштабирования строк, $S = \mathrm{diag}(s_{jj})$ — заданная диагональная матрица масштабирования столбцов. (Все диагональные элементы r_{ii} и s_{jj} предполагаются положительными.)

Примечание. Вычисление указанных масштабирующих матриц R и S представляет собой самостоятельную проблему, которая здесь не рассматривается.

Так как масштабирующие матрицы R и S являются невырожденными, то применяя подходящие тождественные преобразования к исходной (немасштабированной) ЛП-задаче (П.7) будем иметь:

$$\begin{gathered} z = 0^T R^{-1} R x_R + c^T S S^{-1} x_S + c_0 \to \min \\ R x_R = RAS S^{-1} x_S \\ RL \le R x_R \le RU \\ S^{-1} l \le S^{-1} x_S \le S^{-1} u \end{gathered}$$

или с учетом (П.11):

$$\begin{gathered} z = 0^T \widetilde{x}_R + \widetilde{c}^T \widetilde{x}_S + c_0 \to \min \\ \widetilde{x}_R = \widetilde{A} \widetilde{x}_S \\ \widetilde{L} \le \widetilde{x}_R \le \widetilde{U} \\ \widetilde{l} \le \widetilde{x}_S \le \widetilde{u} \end{gathered} \tag{П.12}$$

где величины с тильдой представляют соответствующие компоненты масштабированной ЛП-задачи:

$$\begin{aligned} \widetilde{x}_R &= R x_R, \quad \widetilde{x}_S = S^{-1} x_S, \\ \widetilde{c} &= S x_S, \\ \widetilde{L} &= RL, \qquad \widetilde{U} = RU, \\ \widetilde{l} &= S^{-1} l, \qquad \widetilde{u} = S^{-1} u. \end{aligned} \tag{П.13}$$

После получения промежуточного или окончательного базисного решения масштабированной ЛП-задачи (П.12) необходимо выполнить обратное масштабирование (демасштабирование), чтобы это решение соответствовало исходной немасштабированной ЛП-задаче (П.7). Демасштабирование компонент базисного решения выполняется следующим образом.

Пусть $\widetilde{\beta}_i$ — вычисленное масштабированное значение базисной переменной $(x_B)_i = x_k$. Так как связь между масштабированным и немасштабированным значениями переменной x_k та же, что и между границами этой переменной, то из (П.13) имеем:

$$\beta_i = \begin{cases} (1/r_{ii})\widetilde{\beta}_i, & \text{если } x_k = (x_R)_i \\ s_{jj}\widetilde{\beta}_i, & \text{если } x_k = (x_S)_j \end{cases} \tag{П.14}$$

В случае небазисных переменных демасштабирование их значений не требуется, так как в этом случае можно использовать соответствующие исходные (немасштабированные) активные границы этих переменных.

Аналогично, пусть $\widetilde{d}_j$ — вычисленная масштабированная относительная оценка небазисной переменной $(x_N)_j = x_k$. Связь между масштабированной и немасштабированной относительными оценками та же, что и между коэффициентами целевой функции при этой переменной. Поэтому из (П.13) имеем:

$$d_j = \begin{cases} r_{ii}\widetilde{d}_j, & \text{если } x_k = (x_R)_i \\ (1/s_{jj})\widetilde{d}_j, & \text{если } x_k = (x_S)_j \end{cases} \qquad \text{(П.15)}$$

По определению относительные оценки базисных переменных равны нулю, поэтому демасштабирование таких оценок не требуется.

В заключение отметим, что демасштабирование значения целевой функции в базисном решении также не требуется, поскольку это значение будет одним и тем же как для исходной немасштабированной (П.7), так и для масштабированной (П.12) ЛП-задач.

П.6.3 Исключение небазисных фиксированных переменных

В соответствии с общей схемой симплекс-метода (см. подразд. 1.3) небазисные фиксированные переменные не могут изменяться и поэтому никогда не выбираются для последующего ввода в базис. Это позволяет исключить такие переменные из рабочей ЛП-задачи на этапе формирования исходных данных, поскольку начальный базис предполагается заданным и поскольку исключение соответствующих столбцов из матрицы коэффициентов ограничений не изменяет базисную матрицу.

С точки зрения ЛП-задачи (1.1)—(1.3) исключение небазисной фиксированной переменной состоит в простой подстановке фиксированного значения этой переменной $x_k = l_k (= u_k)$ в целевую функцию (1.1) и ограничения (1.2). Это приводит к соответствующему изменению свободного члена целевой функции c_0:

$$c_0' = c0 + c_k l_k, \qquad \text{(П.16)}$$

где c_k — коэффициент целевой функции при переменной x_k, и вектора правых частей ограничений b:

$$b' = b - A_k l_k, \qquad \text{(П.17)}$$

где A_k — k-й столбец матрицы коэффициентов ограничений A, соответствующий переменной x_k.

Понятно, что значение исключенной фиксированной переменной x_k в базисном решении исходной (включающей эту переменную) ЛП-задачи будет равно $l_k (= u_k)$. Относительную оценку d_k этой переменной можно вычислить, используя формулу (2.15):

$$d_k = c_k - A_k^T \pi, \qquad \text{(П.18)}$$

где c_k — коэффициент целевой функции при x_k, A_k — столбец матрицы коэффициентов ограничений, соответствующий x_k, π — вектор симплексных множителей.

Примечание. Аналогичное исключение базисных свободных (неограниченных) переменных в пакете GLPK не применяется по двум причинам. Во-первых, такое исключение изменило бы базисную матрицу, что повлекло бы за собой необходимость вычисления (или пересчета) ее факторизации. И во-вторых, что более существенно, в реальных ЛП-задачах свободные переменные встречаются достаточно редко.

П.6.4 Смещение границ переменных

При формировании исходных данных ЛП-задачи в пакете GLPK предусмотрена необязательная возможность смещения границ переменных, которая позволяет несколько повысить точность решения.

С точки зрения рабочей ЛП-задачи (1.1)—(1.3) смещение переменной $l_k \le x_k \le u_k$ на ее нижнюю границу l_k (при условии, что нижняя граница существует) представляет собой подстановку:

$$x_k = x'_k + l_k, \tag{П.19}$$

где нижняя граница новой переменной $0 \le x'_k \le u_k - l_k$ становится нулевой. Эта подстановка приводит к изменению свободного члена целевой функции c_0:

$$c'_0 = c0 + c_k l_k, \tag{П.20}$$

где c_k — коэффициент целевой функции при переменной x_k, и вектора правых частей ограничений b:

$$b' = b - A_k l_k, \tag{П.21}$$

где A_k — k-й столбец матрицы коэффициентов ограничений A, соответствующий переменной x_k.

Аналогично, смещение переменной $l_k \le x_k \le u_k$ на ее верхнюю границу u_k (при условии, что верхняя граница существует) представляет собой подстановку:

$$x_k = x'_k + u_k, \tag{П.22}$$

где верхняя граница новой переменной $l_k - u_k \le x'_k \le 0$ становится нулевой. Эта подстановка приводит к соответствующему изменению свободного члена целевой функции и вектора правых частей ограничений:

$$c'_0 = c0 + c_k u_k, \tag{П.23}$$

$$b' = b - A_k u_k. \tag{П.24}$$

Восстановление несмещенных значений переменных в базисном решении выполняется в соответствии с формулами (П.19) и (П.22). При этом достаточно восстанавливать значения только базисных переменных, так как для небазисных переменных можно использовать значения их исходных (несмещенных) активных границ.

Отметим, что смещение границ переменных не изменяет вектор коэффициентов целевой функции $c = (c_k)$ и матрицу коэффициентов ограничений $A = (a_{ik})$, а значит, не влияет на базисную матрицу B и ее факторизацию, и в частности, оставляет неизменными относительные оценки небазисных переменных.

П.6.5 Основные параметры подпрограмм

Во всех подпрограммах данного подраздела параметр `SPXLP *lp` есть указатель на программный объект, представляющий рабочую ЛП-задачу в формате (1.1)—(1.3) (см. подразд. П.1), параметр `glp_prob *P` есть указатель на программный объект, представляющий исходную ЛП-задачу в обобщенном формате GLPK (П.1)—(П.3).

Предполагается, что ЛП-объект `P` содержит хотя бы одну строку (`P->m` > 0), а также определяет корректный текущий базис и имеет действительную факторизацию соответствующей базисной матрицы (установлен флажок `P->valid`).

П.6.6 spx_init_lp — определение размерности рабочей задачи

```
void spx_init_lp(SPXLP *lp, glp_prob *P, int excl);
```

Данная подпрограмма определяет число ограничений-равенств m, число переменных n и число ненулевых элементов матрицы коэффициентов ограничений `nnz` для рабочей ЛП-задачи, которая соответствует исходной ЛП-задаче `P`, и записывает эти величины в объект `lp`. (Память для объекта `lp` должна быть отведена в вызывающей программе.)

Если установлен признак `excl`, то при определении размерности рабочей ЛП-задачи считается, что небазисные фиксированные переменные будут исключены.

Вызов данной подпрограммы должен быть первым в цепочке вызовов подпрограмм, связаных с формированием рабочей ЛП-задачи и ее начального базиса.

П.6.7 spx_alloc_lp — размещение массивов рабочей задачи

```
void spx_alloc_lp(SPXLP *lp);
```

Данная подпрограмма выполняет отведение памяти для всех массивов, входящих в состав объекта `lp`.

Вызов данной подпрограммы должен следовать непосредственно за вызовом подпрограммы `spx_init_lp`.

П.6.8 spx_build_lp — формирование рабочей задачи

```
void spx_build_lp(SPXLP *lp, glp_prob *P, int excl, int shift,
   int map[/*1+P->m+P->n*/]);
```

Данная подпрограмма выполняет преобразование исходной ЛП-задачи `P` (П.1)—(П.3) в рабочую ЛП-задачу `lp` (1.1)—(1.3). Это преобразование включает в себя формирование и масштабирование всех компонент рабочей ЛП-задачи (см. подразд. П.6.1 и П.6.2), за исключением формирования начального базиса, а также, если исходная ЛП-задача имеет вид максимизации, изменение знаков коэффициентов и свободного члена целевой функции на противоположные.

Если установлен признак `excl`, фиксированные небазисные переменные не включаются в рабочую ЛП-задачу (см. подразд. П.6.3). В противном случае, если этот признак не установлен, в рабочую ЛП-задачу включаются все (вспомогательные и структурные) переменные исходной ЛП-задачи.

Примечание. Значение признака `excl` должно быть тем же, что и при вызове подпрограммы `spx_init_lp` (см. выше).

Если установлен признак `shift`, подпрограмма выполняет смещение границ (см. подразд. П.6.4) всех переменных, включаемых в рабочую ЛП-задачу. Если при этом некоторая переменная имеет обе нижнюю и верхнюю границы, смещение выполняется к той границе, которая является меньшей по абсолютной величине.

На выходе подпрограмма формирует массив `map`, который содержит 1+`P->m`+`P->n` ячеек (ячейка с индексом 0 не используется), где `P->m` — число строк и `P->n` — число столбцов в исходной ЛП-задаче. Этот массив устанавливает связь между номерами переменных исходной и рабочей ЛП-задач следующим образом:

`map`$[i] = 0$, $1 \le i \le$ `P->m`, означает, что i-я вспомогательная переменная исходной ЛП-задачи была исключена из рабочей ЛП-задачи;

`map`$[i] = +k$, $1 \le i \le$ `P->m`, означает, что i-я вспомогательная переменная исходной ЛП-задачи соответствует переменной x_k рабочей ЛП-задачи;

`map`$[i] = -k$, $1 \le i \le$ `P->m`, означает, что i-я вспомогательная переменная исходной ЛП-задачи соответствует переменной x_k рабочей ЛП-задачи, причем было выполнено смещение к верхней границе этой переменной;

`map`[`P->m`+j], $1 \le i \le$ `P->n`, имеет тот же смысл, что и выше, но для j-й структурной переменной исходной ЛП-задачи.

Вызов данной подпрограммы должен следовать непосредственно за вызовом подпрограммы `spx_alloc_lp`.

П.6.9 spx_build_basis — формирование базиса рабочей задачи

```
void spx_build_basis(SPXLP *lp, glp_prob *P, const int map[]);
```

Данная подпрограмма формирует начальный базис для рабочей ЛП-задачи `lp`, соответствующий текущему базису исходной ЛП-задачи `P` (информация о текущем базисе является частью программного объекта `P`), а также перемещает драйвер факторизации базисной матрицы из объекта исходной ЛП-задачи в объект рабочей ЛП-задачи.

Массив `map` должен содержать информацию, которая была предварительно записана подпрограммой `spx_build_lp` (см. выше).

Вызов данной подпрограммы завершает цепочку вызовов, связаных с формированием рабочей ЛП-задачи и ее начального базиса, и должен следовать непосредственно за вызовом подпрограммы `spx_build_lp`.

П.6.10 spx_store_basis — формирование базиса исходной задачи

```
void spx_store_basis(SPXLP *lp, glp_prob *P, const int map[],
   int daeh[/*1+n*/]);
```

Используя текущий базис рабочей ЛП-задачи `lp` данная подпрограмма формирует соответствующий базис для исходной ЛП-задачи `P`, что включает в себя определение и запись статусов всех строк (вспомогательных переменных) и всех столбцов (структурных переменных), а также формирование заголовка базиса.

Массив `map` должен содержать информацию, которая была предварительно записана подпрограммой `spx_build_lp` (см. выше).

На выходе подпрограмма формирует массив `daeh`. Этот массив содержит 1+`lp->n` ячеек (ячейка с индексом 0 не используется) и представляет обращение массива `lp->head`, т. е. если `lp->head[k']` $= k$, то `daeh[k]` $= k'$.

П.6.11 spx_store_sol — формирование базисного решения исходной задачи

```
void spx_store_sol(SPXLP *lp, glp_prob *P, int shift, const int map[],
   const int daeh[], const double beta[], const double pi[], const double d[]);
```

Данная подпрограмма выполняет преобразование текущего базисного решения рабочей ЛП-задачи `lp` (значения базисных переменных, симплексные множители и относительные оценки небазисных переменных) в базисное решение исходной ЛП-задачи `P` (значения и относительные оценки вспомогательных и структурных переменных). Это преобразование включает в себя демасштабирование всех компонент базисного решения (см. подразд. П.6.2), вычисление относительных оценок исключенных небазисных переменных (см. подразд. П.6.3), восстановление несмещенных значений базисных переменных (см. подразд. П.6.4), изменение знаков относительных оценок на противоположные (если исходная ЛП-задача имеет вид максимизации), а также вычисление значения целевой функции.

Значение признака `shift` должно совпадать с тем, которое было задано при вызове подпрограммы `spx_build_lp` (см. выше).

Массив `map` должен содержать информацию, которая была предварительно записана подпрограммой `spx_build_lp` (см. выше).

Массив `daeh` должен содержать информацию, которая была сформирована подпрограммой `spx_store_basis` (см. выше).

В ячейках массива `beta[1]`, ..., `beta[m]` должны находиться компоненты вектора значений базисных переменных $\beta = (\beta_i)$ (1.12) для рабочей ЛП-задачи.

В ячейках массива `pi[1]`, ..., `pi[m]` должны находиться компоненты вектора симплексных множителей $\pi = (\pi_i)$ (1.16) для рабочей ЛП-задачи.

В ячейках массива `d[1]`, ..., `d[n-m]` должны находиться компоненты вектора относительных оценок небазисных переменных $d = (d_j)$ (1.15) для рабочей ЛП-задачи.

П.6.12 spx_free_lp — освобождение массивов

```
void spx_free_lp(SPXLP *lp);
```

Данная подпрограмма выполняет освобождение памяти, занятой всеми массивами, входящими в состав объекта `lp`.