Подпрограммы препроцессорной обработки ЛП- и ЦЛП-задач

А. О. Махорин*

Январь 2010 г.

Аннотация

Данный технический меморандум содержит описания подпрограмм, входящих в состав модуля препроцессора пакета GLPK и предназначенных для препроцессорной обработки ЛП- и ЦЛП-задач. Подпрограммы первой группы выполняют элементарные преобразования, которые, в частности, можно использовать для перехода от общего формата задачи к ограниченному формату (если, например, все ограничения должны иметь вид равенств или все переменные должны быть неотрицательными). Подпрограммы второй группы выполняют преобразования, направленные на упрощение исходной задачи. Целочисленные подпрограммы предназначены для выполнения некоторых преобразований целочисленных задач.

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

Содержание

1 Первая группа подпрограмм

1.1 Обработка свободной строки

Спецификация

```
#include "glpnpp.h"
void npp_free_row(NPP *npp, NPPROW *p);
```

Назначение

Подпрограмма `npp_free_row` обрабатывает строку p, которая является свободной (неограниченной по знаку):

$$-\infty < \sum_j a_{pj} x_j < +\infty. \tag{1}$$

Преобразование задачи

Ограничение (1) не может быть активным, следовательно, оно является избыточным и его можно исключить из исходной задачи.

Удаление строки p в прямой системе приводит к удалению столбца для множителя этой строки π_p в двойственной системе. Поскольку строка p не имеет границ, то $\pi_p = 0$, поэтому удаление соответствующего столбца не влияет на двойственное решение.

Восстановление базисного решения

В решении исходной задачи строка p является неактивным ограничением, поэтому она получает статус `GLP_BS`, а ее множитель π_p полагается равным нулю.

Восстановление решения внутренней точки

В решении исходной задачи строка p является неактивным ограничением, поэтому ее множитель π_p полагается равным нулю.

Восстановление целочисленного решения

Не требуется.

1.2 Обработка строки типа «не меньше»

Спецификация

```
#include "glpnpp.h"
void npp_geq_row(NPP *npp, NPPROW *p);
```

Назначение

Подпрограмма `npp_geq_row` обрабатывает строку p, которая является ограничением-неравенством типа «не меньше»:

$$L_p \leq \sum_j a_{pj} x_j \ (\leq U_p), \tag{1}$$

где $L_p < U_p$, причем верхняя граница может отсутствовать ($U_p = +\infty$).

Преобразование задачи

Ограничение (1) можно преобразовать к ограничению-равенству:

$$\sum_j a_{pj} x_j - s = L_p, \tag{2}$$

где

$$0 \leq s \ (\leq U_p - L_p) \tag{3}$$

есть неотрицательная *переменная избытка*.

Так как в прямой системе появляется новый столбец s, имеющий единственный ненулевой коэффициент в строке p, то в двойственной системе появляется новая строка:

$$(-1)\pi_p + \lambda = 0, \tag{4}$$

где (-1) — коэффициент столбца s в строке p, π_p — множитель строки p, λ — множитель столбца s, 0 — коэффициент столбца s в строке целевой функции.

Восстановление базисного решения

Статус строки p в решении исходной задачи определяется статусом этой строки и статусом столбца s в решении преобразованной задачи:

Преобразованная задача		Исходная задача
Статус строки p	Статус столбца s	Статус строки p
GLP_BS	GLP_BS	—
GLP_BS	GLP_NL	GLP_BS
GLP_BS	GLP_NU	GLP_BS
GLP_NS	GLP_BS	GLP_BS
GLP_NS	GLP_NL	GLP_NL
GLP_NS	GLP_NU	GLP_NU

Значение множителя строки π_p в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Примечания:

1. В решении преобразованной задачи строка p и столбец s не могут быть базисными одновременно, так как в противном случае в базисной матрице имелись бы два линейно зависимых столбца — единичный столбец вспомогательной переменной строки p и единичный столбец переменной s.

2. Хотя в преобразованной задаче строка p является ограничением-равенством, она может быть базисной (неактивной) в случае вырожденного решения.

Восстановление решения внутренней точки

Значение множителя строки π_p в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Восстановление целочисленного решения

Не требуется.

1.3 Обработка строки типа «не больше»

Спецификация

```
#include "glpnpp.h"
void npp_leq_row(NPP *npp, NPPROW *p);
```

Назначение

Подпрограмма `npp_leq_row` обрабатывает строку p, которая является ограничением-неравенством типа «не больше»:

$$(L_p \leq) \sum_j a_{pj} x_j \leq U_p, \tag{1}$$

где $L_p < U_p$, причем нижняя граница может отсутствовать ($L_p = -\infty$).

Преобразование задачи

Ограничение (1) можно преобразовать к ограничению-равенству:

$$\sum_j a_{pj} x_j + s = U_p, \tag{2}$$

где

$$0 \leq s \ (\leq U_p - L_p) \tag{3}$$

есть неотрицательная *переменная недостатка*.

Так как в прямой системе появляется новый столбец s, имеющий единственный ненулевой коэффициент в строке p, то в двойственной системе появляется новая строка:

$$(+1)\pi_p + \lambda = 0, \tag{4}$$

где $(+1)$ — коэффициент столбца s в строке p, π_p — множитель строки p, λ — множитель столбца s, 0 — коэффициент столбца s в строке целевой функции.

Восстановление базисного решения

Статус строки p в решении исходной задачи определяется статусом этой строки и статусом столбца s в решении преобразованной задачи:

Преобразованная задача		Исходная задача
Статус строки p	Статус столбца s	Статус строки p
GLP_BS	GLP_BS	—
GLP_BS	GLP_NL	GLP_BS
GLP_BS	GLP_NU	GLP_BS
GLP_NS	GLP_BS	GLP_BS
GLP_NS	GLP_NL	GLP_NU
GLP_NS	GLP_NU	GLP_NL

Значение множителя строки π_p в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Примечания:

1. В решении преобразованной задачи строка p и столбец s не могут быть базисными одновременно, так как в противном случае в базисной матрице имелись бы два линейно зависимых столбца — единичный столбец вспомогательной переменной строки p и единичный столбец переменной s.

2. Хотя в преобразованной задаче строка p является ограничением-равенством, она может быть базисной (неактивной) в случае вырожденного решения.

Восстановление решения внутренней точки

Значение множителя строки π_p в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Восстановление целочисленного решения

Не требуется.

1.4 Обработка свободного столбца

Спецификация

```
#include "glpnpp.h"
void npp_free_col(NPP *npp, NPPCOL *q);
```

Назначение

Подпрограмма `npp_free_col` обрабатывает столбец q, который является свободным (неограниченным по знаку):

$$-\infty < x_q < +\infty. \tag{1}$$

Преобразование задачи

Свободную переменную можно заменить разностью неотрицательных переменных:

$$x_q = s' - s'', \quad s', s'' \geq 0. \tag{2}$$

Если считать, что в преобразованной задаче x_q соответствует s', то преобразование (2) приводит к появлению нового столбца s'', который отличается от столбца s' только знаком коэффициентов в строках ограничений и целевой функции. Таким образом, если в двойственной системе столбцу s' соответствует строка:

$$\sum_i a_{iq}\pi_i + \lambda' = c_q, \tag{3}$$

то столбцу s'' будет соответствовать строка:

$$\sum_i (-a_{iq})\pi_i + \lambda'' = -c_q, \tag{4}$$

откуда следует, что:

$$\lambda' + \lambda'' = 0 \quad \Rightarrow \quad \lambda' = \lambda'' = 0, \tag{5}$$

где $\lambda', \lambda'' \geq 0$ — множители столбцов s' и s''.

Восстановление базисного решения

В соответствии с (5) статус столбца q в решении исходной задачи определяется статусом столбцов s' и s'' в решении преобразованной задачи:

Преобразованная задача		Исходная задача
Статус столбца s'	Статус столбца s''	Статус столбца q
GLP_BS	GLP_BS	—
GLP_BS	GLP_NL	GLP_BS
GLP_NL	GLP_BS	GLP_BS
GLP_NL	GLP_NL	GLP_NF

Значение столбца q вычисляется по формуле (2).

Примечания:

1. В решении преобразованной задачи столбцы s' и s'' не могут быть базисными одновременно, поскольку они отличаются только знаком, а значит, являются линейно зависимыми.

2. Хотя столбец q является свободным, он может быть небазисным в случае вырожденного двойственного решения исходной задачи.

3. Если столбец q является целочисленным, то столбцы s' и s'' также являются целочисленными.

Восстановление решения внутренней точки

Значение столбца q вычисляется по формуле (2).

Восстановление целочисленного решения

Значение столбца q вычисляется по формуле (2).

1.5 Обработка столбца с ненулевой нижней границей

Спецификация

```
#include "glpnpp.h"
void npp_lbnd_col(NPP *npp, NPPCOL *q);
```

Назначение

Подпрограмма `npp_lbnd_col` обрабатывает столбец q, который имеет ненулевую нижнюю границу:

$$l_q \leq x_q \ (\leq u_q), \tag{1}$$

где $l_q < u_q$, причем верхняя граница может отсутствовать ($u_q = +\infty$).

Преобразование задачи

Столбец q можно заменить следующим образом:

$$x_q = l_q + s, \tag{2}$$

где

$$0 \leq s \ (\leq u_q - l_q) \tag{3}$$

является неотрицательной переменной.

Подставляя x_q из (2) в строку целевой функции, получим:

$$\begin{aligned} z &= \sum_j c_j x_j + c_0 = \sum_{j \neq q} c_j x_j + c_q x_q + c_0 = \\ &= \sum_{j \neq q} c_j x_j + c_q (l_q + s) + c_0 = \sum_{j \neq q} c_j x_j + c_q s + \overline{c}_0, \end{aligned}$$

где

$$\overline{c}_0 = c_0 + c_q l_q \tag{4}$$

есть свободный член целевой функции в преобразованной задаче. Аналогично, подставляя x_q в строку i, получим:

$$\begin{aligned} &L_i \leq \sum_j a_{ij} x_j \leq U_i \quad \Rightarrow \quad L_i \leq \sum_{j \neq q} a_{ij} x_j + a_{iq} x_q \leq U_i \quad \Rightarrow \\ &L_i \leq \sum_{j \neq q} a_{ij} x_j + a_{iq} (l_q + s) \leq U_i \quad \Rightarrow \quad \overline{L}_i \leq \sum_{j \neq q} a_{ij} x_j + a_{iq} s \leq \overline{U}_i, \end{aligned}$$

где

$$\overline{L}_i = L_i - a_{iq} l_q, \ \ \overline{U}_i = U_i - a_{iq} l_q \tag{5}$$

есть нижняя и верхняя границы строки i в преобразованной задаче.

Преобразование (2) не изменяет двойственную систему.

Восстановление базисного решения

Статус столбца q в решении исходной задачи совпадает со статусом столбца s в решении преобразованной задачи (`GLP_BS`, `GLP_NL` или `GLP_NU`). Значение столбца q вычисляется по формуле (2).

Восстановление решения внутренней точки

Значение столбца q вычисляется по формуле (2).

Восстановление целочисленного решения

Значение столбца q вычисляется по формуле (2).

1.6 Обработка столбца с верхней границей

Спецификация

```
#include "glpnpp.h"
void npp_ubnd_col(NPP *npp, NPPCOL *q);
```

Назначение

Подпрограмма `npp_ubnd_col` обрабатывает столбец q, который имеет верхнюю границу:

$$(l_q \leq)\ x_q \leq u_q, \tag{1}$$

где $l_q < u_q$, причем нижняя граница может отсутствовать ($l_q = -\infty$).

Преобразование задачи

Столбец q можно заменить следующим образом:

$$x_q = u_q - s, \tag{2}$$

где

$$0 \leq s\ (\leq u_q - l_q) \tag{3}$$

является неотрицательной переменной.

Подставляя x_q из (2) в строку целевой функции, получим:

$$\begin{aligned} z &= \sum_j c_j x_j + c_0 = \sum_{j \neq q} c_j x_j + c_q x_q + c_0 = \\ &= \sum_{j \neq q} c_j x_j + c_q (u_q - s) + c_0 = \sum_{j \neq q} c_j x_j - c_q s + \overline{c}_0, \end{aligned}$$

где

$$\overline{c}_0 = c_0 + c_q u_q \tag{4}$$

есть свободный член целевой функции в преобразованной задаче. Аналогично, подставляя x_q в строку i, получим:

$$\begin{aligned} &L_i \leq \sum_j a_{ij} x_j \leq U_i \quad \Rightarrow \quad L_i \leq \sum_{j \neq q} a_{ij} x_j + a_{iq} x_q \leq U_i \quad \Rightarrow \\ &L_i \leq \sum_{j \neq q} a_{ij} x_j + a_{iq} (u_q - s) \leq U_i \quad \Rightarrow \quad \overline{L}_i \leq \sum_{j \neq q} a_{ij} x_j - a_{iq} s \leq \overline{U}_i, \end{aligned}$$

где

$$\overline{L}_i = L_i - a_{iq} u_q, \quad \overline{U_i} = U_i - a_{iq} u_q \tag{5}$$

есть нижняя и верхняя границы строки i в преобразованной задаче.

Заметим, что при переходе к преобразованной задаче коэффициенты c_q и a_{iq} меняют свои знаки на противоположные. Поэтому строка двойственной системы, соответствующая столбцу q:

$$\sum_i a_{iq}\pi_i + \lambda_q = c_q \tag{6}$$

в преобразованной задаче примет следующий вид:

$$\sum_i (-a_{iq})\pi_i + \lambda_s = -c_q, \tag{7}$$

откуда следует, что:

$$\lambda_q = -\lambda_s, \tag{8}$$

где λ_q — множитель столбца q в исходной задаче, λ_s — множитель столбца s в преобразованной задаче.

Восстановление базисного решения

В соответствии с (8) статус столбца q в решении исходной задачи определяется статусом столбца s в решении преобразованной задачи следующим образом:

Статус столбца s (преобразованная задача)	Статус столбца q (исходная задача)
GLP_BS	GLP_BS
GLP_NL	GLP_NU
GLP_NU	GLP_NL

Значение столбца q вычисляется по формуле (2).

Восстановление решения внутренней точки

Значение столбца q вычисляется по формуле (2).

Восстановление целочисленного решения

Значение столбца q вычисляется по формуле (2).

1.7 Обработка неотрицательного столбца с верхней границей

Спецификация

```
#include "glpnpp.h"
void npp_dbnd_col(NPP *npp, NPPCOL *q);
```

Назначение

Подпрограмма `npp_dbnd_col` обрабатывает столбец q, который является неотрицательным и имеет верхнюю границу:

$$0 \le x_q \le u_q, \tag{1}$$

где $u_q > 0$.

Преобразование задачи

Верхнюю границу столбца q можно заменить следующим ограничением-равенством:

$$x_q + s = u_q, \tag{2}$$

где $s \ge 0$ — неотрицательная *переменная дополнения*.

Так как в прямой системе вместе с новой строкой (2) появляется новый столбец s, имеющий единственный ненулевой коэффициент в этой строке, то в двойственной системе появляется новая строка:

$$(+1)\pi + \lambda_s = 0, \tag{3}$$

где $(+1)$ — коэффициент столбца s в строке (2), π — множитель строки (2), λ_s — множитель столбца s, 0 — коэффициент столбца s в строке целевой функции.

Восстановление базисного решения

Статус столбца q в решении исходной задачи определяется его статусом и статусом столбца s в решении преобразованной задачи:

Преобразованная задача		Исходная задача
Статус столбца q	Статус столбца s	Статус столбца q
GLP_BS	GLP_BS	GLP_BS
GLP_BS	GLP_NL	GLP_NU
GLP_NL	GLP_BS	GLP_NL
GLP_NL	GLP_NL	GLP_NL*

Значение столбца q в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Примечания:

1. Формально в решении преобразованной задачи столбцы q и s не могут быть небазисными одновременно, так как в противном случае было бы нарушено ограничение (2). Однако, если величина u_q близка к нулю, то нарушение ограничения (2) может находиться в пределах допустимой погрешности даже если оба столбца q и s небазисные. В этом вырожденном случае строка (2) может быть только базисной, т. е. быть только неактивным ограничением (в противном случае соответствующая строка базисной матрицы была бы нулевой). Это позволяет вывести из базиса вспомогательную переменную строки (2), заменяя ее переменной дополнения s, и считать, таким образом, что столбец s является базисным.

2. Если столбец q является целочисленным, то столбец s также является целочисленными.

Восстановление решения внутренней точки

Значение столбца q в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Восстановление целочисленного решения

Значение столбца q в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

1.8 Обработка фиксированного столбца

Спецификация

```
#include "glpnpp.h"
void npp_fixed_col(NPP *npp, NPPCOL *q);
```

Назначение

Подпрограмма `npp_fixed_col` обрабатывает столбец q, который является фиксированным:

$$x_q = s_q, \tag{1}$$

где s_q — соответствующее значение столбца.

Преобразование задачи

Значение фиксированного столбца можно подставить в строку целевой функции и строки ограничений и тем самым исключить этот столбец из задачи.

Подставляя $x_q = s_q$ в строку целевой функции, получим:

$$\begin{aligned} z &= \sum_j c_j x_j + c_0 = \sum_{j \neq q} c_j x_j + c_q x_q + c_0 = \\ &= \sum_{j \neq q} c_j x_j + c_q s_q + c_0 = \sum_{j \neq q} c_j x_j + \overline{c}_0, \end{aligned}$$

где

$$\overline{c}_0 = c_0 + c_q s_q \tag{2}$$

есть свободный член целевой функции в преобразованной задаче. Аналогично, подставляя $x_q = s_q$ в строку i, получим:

$$\begin{aligned} L_i \leq \sum_j a_{ij} x_j \leq U_i \quad &\Rightarrow \quad L_i \leq \sum_{j \neq q} a_{ij} x_j + a_{iq} x_q \leq U_i \quad \Rightarrow \\ L_i \leq \sum_{j \neq q} a_{ij} x_j + a_{iq} s_q \leq U_i \quad &\Rightarrow \quad \overline{L}_i \leq \sum_{j \neq q} a_{ij} x_j \leq \overline{U}_i, \end{aligned}$$

где

$$\overline{L}_i = L_i - a_{iq} s_q, \quad \overline{U}_i = U_i - a_{iq} s_q \tag{5}$$

есть нижняя и верхняя границы строки i в преобразованной задаче.

Восстановление базисного решения

В решении исходной задачи столбец q получает статус `GLP_NS`, а его значение полагается равным s_q.

Восстановление решения внутренней точки

Значение столбца q полагается равным s_q.

Восстановление целочисленного решения

Значение столбца q полагается равным s_q.

1.9 Обработка почти равных границ строки

Спецификация

```
#include "glpnpp.h"
int npp_make_equality(NPP *npp, NPPROW *p);
```

Назначение

Подпрограмма `npp_make_equality` обрабатывает строку p:

$$L_p \le \sum_j a_{pj} x_j \le U_p, \tag{1}$$

где $-\infty < L_p < U_p < +\infty$, т. е. которая является двусторонним ограничением-неравенством.

Возвращаемое значение

0 — границы строки не изменились;
1 — строка была заменена ограничением-равенством.

Преобразование задачи

Если границы строки (1) очень близки друг к другу:

$$U_p - L_p \le \varepsilon, \tag{2}$$

где ε — абсолютный допуск на значение строки, то такую строку можно заменить почти эквивалентным ограничением-равенством:

$$\sum_j a_{pj} x_j = b. \tag{3}$$

где $b = (L_p + U_p)/2$. Если при этом правая часть (3) оказывается очень близкой к своему ближайшему целому:

$$|b - \lfloor b + 0.5 \rfloor| \le \varepsilon, \tag{4}$$

то целесообразно в качестве правой части использовать это ближайшее целое.

Восстановление базисного решения

Статус строки p в решении исходной задачи определяется ее статусом и знаком ее множителя π_p в решении преобразованной задачи:

Статус строки p (преобразованная задача)	Знак π_p	Статус строки p (исходная задача)
GLP_BS	$\pm$	GLP_BS
GLP_NS	$+$	GLP_NL
GLP_NS	$-$	GLP_NU

Значение множителя строки π_p в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Восстановление решения внутренней точки

Значение множителя строки π_p в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Восстановление целочисленного решения

Не требуется.

1.10 Обработка почти равных границ столбца

Спецификация

```
#include "glpnpp.h"
int npp_make_fixed(NPP *npp, NPPCOL *q);
```

Назначение

Подпрограмма `npp_make_fixed` обрабатывает столбец q:

$$l_q \le x_q \le u_q, \tag{1}$$

где $-\infty < l_q < u_q < +\infty$, т. е. который имеет обе нижнюю и верхнюю границы.

Возвращаемое значение

0 — границы столбца не изменились;
1 — столбец был зафиксирован.

Преобразование задачи

Если границы строки (1) очень близки друг к другу:

$$u_q - l_q \le \varepsilon, \tag{2}$$

где ε — абсолютный допуск на значение столбца, то такой столбец можно зафиксировать:

$$x_q = s_q. \tag{3}$$

где $s_q = (l_q + u_q)/2$. Если при этом фиксированное значение столбца s_q оказывается очень близким к своему ближайшему целому:

$$|s_q - \lfloor s_q + 0.5 \rfloor| \le \varepsilon, \tag{4}$$

то целесообразно в качестве фиксированного значения столбца использовать это ближайшее целое.

Восстановление базисного решения

В двойственной системе как исходной, так и преобразованной задачи столбцу q соответствует следующая строка:

$$\sum_i a_{iq}\pi_i + \lambda_q = c_q. \tag{5}$$

Поскольку значения множителей π_i для всех строк известны из решения преобразованной задачи, равенство (5) позволяет вычислить значение множителя (относительную оценку) для столбца q:

$$\lambda_q = c_q - \sum_i a_{iq}\pi_i. \tag{6}$$

Статус столбца q в решении исходной задачи определяется его статусом знаком его множителя λ_q в решении преобразованной задачи:

Статус столбца q (преобразованная задача)	Знак λ_q	Статус столбца q (исходная задача)
GLP_BS	$\pm$	GLP_BS
GLP_NS	$+$	GLP_NL
GLP_NS	$-$	GLP_NU

Значение столбца q в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Восстановление решения внутренней точки

Значение столбца q в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Восстановление целочисленного решения

Не требуется.

2 Вторая группа подпрограмм

2.1 Обработка пустой строки

Спецификация

```
#include "glpnpp.h"
int npp_empty_row(NPP *npp, NPPROW *p);
```

Назначение

Подпрограмма `npp_empty_row` обрабатывает строку p, которая является пустой, т. е. коэффициенты всех столбцов в этой строке равны нулю:

$$L_p \le \sum_j 0 \cdot x_j \le U_p, \tag{1}$$

где $L_p \le U_p$.

Возвращаемое значение

0 — успешная обработка;
1 — задача не имеет прямых допустимых решений.

Преобразование задачи

Если выполнены условия:

$$L_p \le +\varepsilon, \quad U_p \ge -\varepsilon, \tag{2}$$

где ε — абсолютный допуск на значение строки, то такая строка является избыточной. В этом случае ее можно заменить *эквивалентной* избыточной строкой, которая является свободной (неограниченной по знаку), и исключить из задачи. Если же указанные условия не выполняются, то строка p является недопустимой и, следовательно, задача не имеет (прямых) допустимых решений.

Восстановление базисного решения

См. подразд. «Обработка свободной строки».

Восстановление решения внутренней точки

См. подразд. «Обработка свободной строки».

Восстановление целочисленного решения

Не требуется.

2.2 Обработка пустого столбца

Спецификация

```
#include "glpnpp.h"
int npp_empty_col(NPP *npp, NPPCOL *q);
```

Назначение

Подпрограмма `npp_empty_col` обрабатывает столбец q:

$$l_q \leq x_q \leq u_q, \tag{1}$$

где $l_q < u_q$, который является пустым, т. е. входит во все строки ограничений с нулевым коэффициентом.

Возвращаемое значение

0 — успешная обработка;
1 — задача не имеет двойственных допустимых решений.

Преобразование задачи

Пустому столбцу q в двойственной системе соответствует строка:

$$\sum_i 0 \cdot \pi_i + \lambda_q = c_q, \tag{2}$$

откуда следует, что:

$$\lambda_q = c_q. \tag{3}$$

Если имеет место случай:

$$c_q < -\varepsilon, \tag{4}$$

где ε — абсолютный допуск на значение множителя столбца, то нижняя граница столбца l_q должна быть активной, чтобы обеспечить двойственную допустимость решения.[1] В этом случае столбец можно зафиксировать на нижней границе[2] и исключить его из задачи. Если при этом столбец не имеет конечной нижней границы ($l_q = -\infty$), то задача не имеет двойственных допустимых решений.[3]

[1] Напомним, что во время препроцессорной обработки задача всегда имеет вид минимизации.

[2] Если столбец является целочисленным, то его границы также предполагаются целочисленными.

[3] Это справедливо как для ЛП- так и для ЦЛП-задач. Если ЛП-релаксация не имеет двойственных допустимых решений, то либо она не имеет прямых допустимых решений, и следовательно, ЦЛП-задача также не имеет допустимых решений, либо ЛП-релаксация имеет неограниченный минимум. В последнем случае, если ЦЛП-задача имеет хотя бы одно допустимое целочисленное решение, она также имеет неограниченный (целочисленный) минимум.

Если имеет место случай:

$$c_q > +\varepsilon, \tag{5}$$

то чтобы обеспечить двойственную допустимость решения активной должна быть верхняя граница столбца u_q. В этом случае столбец можно зафиксировать на верхней границе[2] и исключить его из задачи. Если при этом столбец не имеет верхней границы ($u_q = +\infty$), то задача не имеет двойственных допустимых решений.[3]

Если же имеет место случай:

$$-\varepsilon \leq c_q \leq +\varepsilon, \tag{6}$$

то двойственная допустимость решения не зависит от значения столбца. В этом случае столбец можно зафиксировать на нижней (если $l_q > -\infty$) или верхней границе (если $u_q < +\infty$) или в нуле (если столбец не ограничен по знаку) и исключить его из задачи.[4]

Восстановление базисного решения

См. подразд. «Обработка фиксированного столбца». После восстановления столбец будет иметь статус `GLP_NS`. Если на самом деле этот столбец не является фиксированным ($l_q < u_q$), его статус следует заменить на `GLP_NL`, `GLP_NU` или `GLP_NF` в зависимости от того, на какой границе он был зафиксирован при преобразовании задачи.

Восстановление решения внутренней точки

См. подразд. «Обработка фиксированного столбца».

Восстановление целочисленного решения

См. подразд. «Обработка фиксированного столбца».

[4]Если столбец имеет обе нижнюю и верхнюю границы, можно выбрать ту из них, абсолютная величина которой меньше.

2.3 Обработка неявного значения столбца

Спецификация

```
#include "glpnpp.h"
int npp_implied_value(NPP *npp, NPPCOL *q, double s);
```

Назначение

Для столбца q:

$$l_q \leq x_q \leq u_q, \tag{1}$$

где $l_q < u_q$, подпрограмма `npp_implied_value` обрабатывает его заданное неявное значение s_q. Если это значение не противоречит текущим границам столбца и требованиям целочисленности, подпрограмма фиксирует столбец q в точке s_q, при этом столбец остается в задаче.

Возвращаемое значение

0 — столбец зафиксирован в заданной точке;
1 — неявное значение противоречит границам столбца;
2 — неявное значение противоречит требованиям целочисленности.

Алгоритм

Считается, что неявное значение s_q не противоречит границам столбца (1), если выполнено условие:

$$l_q - \varepsilon \leq s_q \leq u_q + \varepsilon, \tag{2}$$

а для целочисленного столбца также условие:

$$|s_q - \lfloor s_q + 0.5 \rfloor| \leq \varepsilon, \tag{3}$$

где ε — абсолютный допуск на значение столбца, $\lfloor s_q + 0.5 \rfloor$ — ближайшее целое к s_q.

Если указанные условия выполнены, то столбец q можно зафиксировать в точке s_q[5] или, если этот столбец является целочисленным, в точке $\lfloor s_q + 0.5 \rfloor$, поскольку значение целочисленного столбца в решении ЦЛП-задачи обязательно должно быть целочисленным. Если же неявное значение s_q не удовлетворяет указанным условиям, то задача не имеет (прямых) допустимых решений.

[5]Если s_q близкó к l_q (u_q), т. е. если, например, выполняется условие $s_q \leq l_q + 0.1\varepsilon$ ($s_q \geq u_q - 0.1\varepsilon$), и при этом граница l_q (u_q) является оригинальной границей столбца, то, вероятно, целесообразнее фиксировать столбец на этой границе, а не в точке s_q.

2.4 Обработка строчного синглета (равенство)

Спецификация

```
#include "glpnpp.h"
int npp_eq_singlet(NPP *npp, NPPROW *p);
```

Назначение

Подпрограмма `npp_eq_singlet` обрабатывает строку p, которая является ограничением-равенством и имеет ровно один ненулевой коэффициент:

$$a_{pq}x_q = b. \tag{1}$$

Возвращаемое значение

0 — успешная обработка;
1 — задача не имеет (прямых) допустимых решений;
2 — задача не имеет целочисленных допустимых решений.

Преобразование задачи

Ограничение-равенство (1) определяет неявное значение столбца q:

$$x_q = s_q = b/a_{pq}. \tag{2}$$

Если s_q не противоречит текущим границам столбца q (см. подразд. «Обработка неявного значения столбца»), этот столбец можно зафиксировать в указанной точке и исключить из задачи. После фиксации столбца q строка p становится избыточной, поэтому ее можно заменить *эквивалентной* свободной строкой и также исключить из задачи.

Примечание. Подпрограмма исключает из задачи только строку p. Столбец q становится фиксированным, но он остается в задаче.

Восстановление базисного решения

В решении исходной задачи строка p получает статус `GLP_NS` (активное ограничение-равенство), а столбец q — статус `GLP_BS` (базисный столбец).

Множитель строки π_p можно вычислить следующим образом. Столбцу q в двойственной системе (исходной задачи) соответствует строка:

$$\sum_i a_{iq}\pi_i + \lambda_q = c_q \quad \Rightarrow \quad \sum_{i \neq p} a_{iq}\pi_i + a_{pq}\pi_p + \lambda_q = c_q,$$

поэтому:

$$\pi_p = \frac{1}{a_{pq}}(c_q - \lambda_q - \sum_{i \neq p} a_{iq}\pi_i), \tag{3}$$

где $\lambda_q = 0$ (так как столбец q — базисный), а значения π_i для всех $i \neq p$ известны из решения преобразованной задачи.

Значением столбца q в решении исходной задачи становится его неявное значение s_q.

Восстановление решения внутренней точки

Для вычисления множителя строки p используется формула (3), а значение столбца q полагается равным его неявному значению s_q.

Восстановление целочисленного решения

Значение столбца q полагается равным его неявному значению s_q.

2.5 Обработка неявной нижней границы столбца

Спецификация

```
#include "glpnpp.h"
int npp_implied_lower(NPP *npp, NPPCOL *q, double l);
```

Назначение

Для столбца q:

$$l_q \leq x_q \leq u_q, \tag{1}$$

где $l_q < u_q$, подпрограмма `npp_implied_lower` обрабатывает его неявную нижнюю границу l_q^*. Результатом может быть увеличение текущей нижней границы столбца l_q, при этом столбец остается в задаче.

Возвращаемое значение

0 — текущая нижняя граница столбца не изменилась;
1 — текущая нижняя граница столбца увеличилась, но незначительно;
2 — текущая нижняя граница столбца значительно увеличилась;
3 — столбец был зафиксирован на своей верхней границе;
4 — неявная граница противоречит текущей верхней границе столбца.

Алгоритм

Если столбец q является целочисленным, то перед проверкой его неявной нижней границы l_q^* ее следует округлить в бóльшую сторону:

$$l_q^* := \begin{cases} \lfloor l_q^* + 0.5 \rfloor, & \text{если } |l_q^* - \lfloor l_q^* + 0.5 \rfloor| \leq \varepsilon \\ \lceil l_q^* \rceil, & \text{в противном случае} \end{cases} \tag{2}$$

где $\lfloor l_q^* + 0.5 \rfloor$ — ближайшее целое к l_q^*; $\lceil l_q^* \rceil$ — наименьшее целое, не меньшее l_q^*; ε — абсолютный допуск на значение столбца.

Обработка неявной нижней границы столбца l_q^* включает следующие случаи:

1) если $l_q^* < l_q + \varepsilon$, то l_q^* является избыточной;

2) если $l_q + \varepsilon \leq l_q^* \leq u_q + \varepsilon$, то текущую нижнюю границу столбца l_q можно увеличить, заменяя ее на l_q^*. Если при этом новая нижняя граница столбца оказалась близкой к его текущей верхней границе u_q, столбец можно зафиксировать на верхней границе;

3) если $l_q^* > u_q + \varepsilon$, то l_q^* противоречит текущей верхней границе столбца u_q, а значит, задача не имеет (прямых) допустимых решений.

2.6 Обработка неявной верхней границы столбца

Спецификация

```
#include "glpnpp.h"
int npp_implied_upper(NPP *npp, NPPCOL *q, double u);
```

Назначение

Для столбца q:

$$l_q \le x_q \le u_q, \tag{1}$$

где $l_q < u_q$, подпрограмма `npp_implied_upper` обрабатывает его неявную верхнюю границу u_q^*. Результатом может быть уменьшение текущей верхней границы столбца u_q, при этом столбец остается в задаче.

Возвращаемое значение

0 — текущая верхняя граница столбца не изменилась;
1 — текущая верхняя граница столбца уменьшилась, но незначительно;
2 — текущая верхняя граница столбца значительно уменьшилась;
3 — столбец был зафиксирован на своей нижней границе;
4 — неявная граница противоречит текущей нижней границе столбца.

Алгоритм

Если столбец q является целочисленным, то перед проверкой его неявной верхней границы u_q^* ее следует округлить в меньшую сторону:

$$u_q^* := \begin{cases} \lfloor u_q^* + 0.5 \rfloor, & \text{если } |u_q^* - \lfloor u_q^* + 0.5 \rfloor| \le \varepsilon \\ \lfloor u_q^* \rfloor, & \text{в противном случае} \end{cases} \tag{2}$$

где $\lfloor u_q^* + 0.5 \rfloor$ — ближайшее целое к u_q^*; $\lfloor u_q^* \rfloor$ — наибольшее целое, не превосходящее u_q^*; ε — абсолютный допуск на значение столбца.

Обработка неявной верхней границы столбца u_q^* включает следующие случаи:

1) если $u_q^* > u_q - \varepsilon$, то u_q^* является избыточной;

2) если $l_q - \varepsilon \le u_q^* \le u_q - \varepsilon$, то текущую верхнюю границу столбца u_q можно усилить, заменяя ее на u_q^*. Если при этом новая верхняя граница столбца оказалась близкой к его текущей нижней границе l_q, столбец можно зафиксировать на нижней границе;

3) если $u_q^* < l_q - \varepsilon$, то u_q^* противоречит текущей нижней границе столбца l_q, а значит, задача не имеет (прямых) допустимых решений.

2.7 Обработка строчного синглета (неравенство)

Спецификация

```
#include "glpnpp.h"
int npp_ineq_singlet(NPP *npp, NPPROW *p);
```

Назначение

Подпрограмма `npp_ineq_singlet` обрабатывает строку p, которая является ограничением-неравенством и имеет ровно один ненулевой коэффициент:

$$L_p \le a_{pq} x_q \le U_p, \tag{1}$$

где $L_p < U_p$, $L_p > -\infty$ и (или) $U_p < +\infty$.

Возвращаемое значение

0 — текущие границы столбца не изменились;
1 — текущие границы столбца изменились, но незначительно;
2 — текущие границы столбца значительно изменились;
3 — столбец был зафиксирован на своей нижней или верхней границе;
4 — задача не имеет (прямых) допустимых решений.

Преобразование задачи

Ограничение-неравенство (1) определяет неявные границы столбца q:

$$l_q^* = \begin{cases} L_p/a_{pq}, & \text{если } a_{pq} > 0 \\ U_p/a_{pq}, & \text{если } a_{pq} < 0 \end{cases} \tag{2}$$

$$u_q^* = \begin{cases} U_p/a_{pq}, & \text{если } a_{pq} > 0 \\ L_p/a_{pq}, & \text{если } a_{pq} < 0 \end{cases} \tag{3}$$

Если эти неявные границы не находятся в противоречии с текущими границами столбца q:

$$l_q \le x_q \le u_q, \tag{4}$$

то их можно использовать для возможного усиления текущих границ:

$$l_q := \max(l_q, l_q^*), \tag{5}$$

$$u_q := \min(u_q, u_q^*). \tag{6}$$

(См. подразд. «Обработка неявной нижней границы столбца» и «Обработка неявной верхней границы столбца».)

После переноса границ строки p на столбец q эта строка становится избыточной, поэтому ее можно заменить *эквивалентной* свободной строкой и исключить из задачи.

Примечание. Подпрограмма исключает из задачи только строку p. Если столбец q становится фиксированным, он все равно остается в задаче.

Восстановление базисного решения

Вначале заметим, что столбцу q в двойственной системе (исходной задачи) соответствует строка:

$$\sum_i a_{iq}\pi_i + \lambda_q = c_q \quad \Rightarrow \quad \sum_{i \neq p} a_{iq}\pi_i + a_{pq}\pi_p + \lambda_q = c_q, \tag{7}$$

где значения π_i для всех $i \neq p$ известны из решения преобразованной задачи. В преобразованной задаче строка p отсутствует, поэтому формально ее множитель там равен нулю. Это позволяет вычислить множитель столбца q в решении *преобразованной* задачи:

$$\overline{\lambda}_q = c_q - \sum_{i \neq p} a_{iq}\pi_i. \tag{8}$$

Пусть в решении преобразованной задачи столбец q является небазисным с активной нижней границей (`GLP_NL`, $\overline{\lambda}_q \geq 0$), причем эта нижняя граница является неявной (2). С точки зрения исходной задачи это означает, что на самом деле исходная нижняя граница столбца (4) неактивна, а активной является та граница ограничения (1), которая определяет неявную нижнюю границу столбца (2). В этом случае в решении исходной задачи столбец q получает статус `GLP_BS`, а строка p — статус `GLP_NL` (если $a_{pq} > 0$) или `GLP_NU` (если $a_{pq} < 0$). Так как теперь столбец q базисный, его множитель равен нулю. Это позволяет использовать равенство (7) для вычисления множителя строки p в решении исходной задачи:

$$\pi_p = \frac{1}{a_{pq}}(c_q - \sum_{i \neq p} a_{iq}\pi_i) = \frac{1}{a_{pq}}\overline{\lambda}_q. \tag{9}$$

Пусть теперь в решении преобразованной задачи столбец q является небазисным с активной верхней границей (`GLP_NU`, $\overline{\lambda}_q \leq 0$), причем эта верхняя граница является неявной (3). Аналогично предыдущему случаю с точки зрения исходной задачи это означает, что на самом деле исходная верхняя граница столбца (4) неактивна, а активной является та граница ограничения (1), которая определяет неявную верхнюю границу столбца (3). В этом случае в решении исходной задачи столбец q получает статус `GLP_BS`, а строка p — статус `GLP_NU` (если $a_{pq} > 0$) или `GLP_NL` (если $a_{pq} < 0$). При этом множитель строки p вычисляется по формуле (9).

Усиление границ столбца q в соответствии с (5) и (6) может привести к фиксации этого столбца. Поэтому если в решении преобразованной задачи фиксированный столбец q оказался небазисным (`GLP_NS`), мы

можем считать, что если $\overline{\lambda}_q > 0$, то этот столбец имеет активную нижнюю границу (GLP_NL), а если $\overline{\lambda}_q < 0$, то этот столбец имеет активную верхнюю границу (GLP_NU), сводя данный случай к двум предыдущим случаям. Если же множитель столбца q в решении преобразованной задачи близок к нулю ($\overline{\lambda}_q \approx 0$) или соответствующая граница строки p отсутствует,[6] то столбцу q можно присвоить статус GLP_BS (базисный), а строку p сделать активной на любой из существующих границ. Поскольку в этом вырожденном случае множитель строки π_p, вычисленный по формуле (9), также будет близок к нулю, двойственная допустимость решения практически не нарушится.

Во всех остальных случаях, а именно, когда в решении преобразованной задачи столбец q является базисным (GLP_BS), небазисным с активной *исходной* нижней границей (GLP_NL), или небазисным с активной *исходной* верхней границей (GLP_NU), ограничение (1) неактивно. Поэтому в решении исходной задачи статус столбца q не изменяется, строка p получает статус GLP_BS, а ее множитель π_p полагается равным нулю.

Восстановление решения внутренней точки

Сначала вычисляется множитель столбца q в решении преобразованной задачи по формуле (8). Если $\overline{\lambda}_q > 0$ и нижняя граница столбца является неявной, или если $\overline{\lambda}_q < 0$ и верхняя граница столбца является неявной, то с точки зрения исходной задачи это означает, что на самом деле активной является соответствующая граница строки p, поэтому в этом случае ее множитель π_p вычисляется по формуле (9). В остальных случаях, когда знак $\overline{\lambda}_q$ соответствует исходной активной границе столбца, или когда $\overline{\lambda}_q \approx 0$, множитель строки π_p полагается равным нулю.

Восстановление целочисленного решения

Не требуется.

[6] Это возможно, если из-за ошибок огрубления множитель $\overline{\lambda}_q$ имеет неправильный знак. Следует заметить, что тогда значение этого множителя должен быть близкó к нулю, так как решение преобразованной задачи предполагается двойственно допустимым.

2.8 Обработка столбцового синглета (неявная переменная недостатка)

Спецификация

```
#include "glpnpp.h"
void npp_implied_slack(NPP *npp, NPPCOL *q);
```

Назначение

Подпрограмма `npp_implied_slack` обрабатывает столбец q:

$$l_q \leq x_q \leq u_q, \tag{1}$$

где $l_q < u_q$, имеющий ненулевой коэффициент лишь в одной строке p, которая является ограничением-равенством:

$$\sum_{j \neq q} a_{pj} x_j + a_{pq} x_q = b. \tag{2}$$

Преобразование задачи[7]

Член $a_{pq}x_q$ в ограничении (2) можно рассматривать как неявную переменную недостатка, что позволяет перенести границы столбца q на строку p и затем исключить этот столбец из задачи.

Запишем ограничение (2) в виде:

$$\sum_{j \neq q} a_{pj} x_j = b - a_{pq} x_q. \tag{3}$$

Из (1) следует, что (3) можно заменить эквивалентным ограничением:

$$L_p \leq \sum_{j \neq q} a_{pj} x_j \leq U_p, \tag{4}$$

где:

$$L_p = \begin{cases} b - a_{pq} u_q, & \text{если } a_{pq} > 0 \\ b - a_{pq} l_q, & \text{если } a_{pq} < 0 \end{cases} \tag{5}$$

$$U_p = \begin{cases} b - a_{pq} l_q, & \text{если } a_{pq} > 0 \\ b - a_{pq} u_q, & \text{если } a_{pq} < 0 \end{cases} \tag{6}$$

Чтобы исключить столбец q из задачи, воспользуемся равенством (2), из которого следует, что:

$$x_q = \frac{1}{a_{pq}} (b - \sum_{j \neq q} a_{pj} x_j). \tag{7}$$

[7] Если столбец q является целочисленным, данное преобразование не применимо.

Подставляя x_q из (7) в строку целевой функции, получим:

$$z = \sum_j c_j x_j + c_0 = \sum_{j \neq q} c_j x_j + c_q x_q + c_0 =$$
$$= \sum_{j \neq q} c_j x_j + c_q \left[\frac{1}{a_{pq}} (b - \sum_{j \neq q} a_{pj} x_j) \right] + c_0 = \sum_{j \neq q} \overline{c}_j x_j + \overline{c}_0,$$

где

$$\overline{c}_j = c_j - c_q \frac{a_{pj}}{a_{pq}}, \quad \overline{c}_0 = c_0 + c_q \frac{b}{a_{pq}} \tag{8}$$

есть значения коэффициентов и свободного члена целевой функции в преобразованной задаче.

Заметим, что столбец q является синглетом, поэтому в двойственной системе исходной задачи ему соответствует строка:

$$a_{pq} \pi_p + \lambda_q = c_q, \tag{9}$$

которая в преобразованной задаче была бы следующей:

$$a_{pq} \overline{\pi}_p + \lambda_q = \overline{c}_q = 0. \tag{10}$$

Вычитая из равенства (9) равенство (10), имеем:

$$a_{pq} (\pi_p - \overline{\pi}_p) = c_q,$$

что позволяет получить формулу для вычисления значения множителя строки p в решении исходной задачи по его значению в решении преобразованной задачи:

$$\pi_p = \overline{\pi}_p + c_q / a_{pq}. \tag{11}$$

Восстановление базисного решения

Статус столбца q в решении исходной задачи определяется статусом строки p в решении преобразованной задачи и знаком коэффициента a_{pq} в исходном ограничении-равенстве (2):

Статус строки p (преобразованная задача)	Знак a_{pq}	Статус столбца q (исходная задача)
GLP_BS	$\pm$	GLP_BS
GLP_NL	$+$	GLP_NU
GLP_NL	$-$	GLP_NL
GLP_NU	$+$	GLP_NL
GLP_NU	$-$	GLP_NU
GLP_NF	$\pm$	GLP_NF

Значение столбца q вычисляется по формуле (7). Так как в исходной задаче строка p является ограничением-равенством, то в решении исходной задачи она получает статус GLP_NS, при этом значение множителя этой строки π_p вычисляется по формуле (11).

Восстановление решения внутренней точки

Значение столбца q вычисляется по формуле (7). Значение множителя строки π_p вычисляется по формуле (11).

Восстановление целочисленного решения

Значение столбца q вычисляется по формуле (7).

2.9 Обработка столбцового синглета (неявная свободная переменная)

Спецификация

```
#include "glpnpp.h"
int npp_implied_free(NPP *npp, NPPCOL *q);
```

Назначение

Подпрограмма `npp_implied_free` обрабатывает столбец q:

$$l_q \le x_q \le u_q, \tag{1}$$

имеющий ненулевой коэффициент лишь в одной строке p, которая является ограничением-неравенством:

$$L_p \le \sum_{j \ne q} a_{pj} x_j + a_{pq} x_q \le U_p, \tag{2}$$

где $l_q < u_q$, $L_p < U_p$, $L_p > -\infty$ и (или) $U_p < +\infty$.

Возвращаемое значение

0 — успешная обработка;
1 — нижняя и (или) верхняя граница столбца может быть активной;
2 — задача не имеет двойственных допустимых решений.

Преобразование задачи

Ограничение (2) можно записать в виде:

$$L_p - \sum_{j \ne q} a_{pj} x_j \le a_{pq} x_q \le U_p - \sum_{j \ne q} a_{pj} x_j,$$

откуда следует, что:

$$\alpha \le a_{pq} x_q \le \beta, \tag{3}$$

где

$$\begin{aligned} \alpha &= \inf(L_p - \sum_{j \ne q} a_{pj} x_j) = L_p - \sup \sum_{j \ne q} a_{pj} x_j = \\ &= L_p - \sum_{j \in J^+} a_{pj} u_j - \sum_{j \in J^-} a_{pj} l_j, \end{aligned} \tag{4}$$

$$\begin{aligned} \beta &= \sup(L_p - \sum_{j \ne q} a_{pj} x_j) = L_p - \inf \sum_{j \ne q} a_{pj} x_j = \\ &= L_p - \sum_{j \in J^+} a_{pj} l_j - \sum_{j \in J^-} a_{pj} u_j, \end{aligned} \tag{5}$$

$$J^+ = \{j \neq q : a_{pj} > 0\}, \quad J^- = \{j \neq q : a_{pj} < 0\}. \tag{6}$$

Неравенство (3) определяет неявные границы переменной x_q:

$$l_q^* \leq x_q \leq u_q^*, \tag{7}$$

где

$$l_q^* = \begin{cases} \alpha/a_{pq}, & \text{если } a_{pq} > 0 \\ \beta/a_{pq}, & \text{если } a_{pq} < 0 \end{cases} \qquad u_q^* = \begin{cases} \beta/a_{pq}, & \text{если } a_{pq} > 0 \\ \alpha/a_{pq}, & \text{если } a_{pq} < 0 \end{cases} \tag{8}$$

Поэтому если $l_q^* > l_q - \varepsilon$ и $u_q^* < u_q + \varepsilon$, где ε — абсолютный допуск на значение столбца, границы столбца (1) не могут быть активными, следовательно, столбец q в этом случае можно считать свободным (неограниченным по знаку).

Так как столбец q является синглетом, в двойственной системе исходной задаче ему соответствует строка:

$$a_{pq}\pi_p + \lambda_q = c_q, \tag{9}$$

откуда:

$$\pi_p = (c_q - \lambda_q)/a_{pq}. \tag{10}$$

Пусть x_q — неявная свободная переменная. Тогда столбец q может быть только базисным, а значит, его множитель λ_q будет равен нулю. Но тогда из (10) следует, что:

$$\pi_p = c_q/a_{pq}, \tag{11}$$

и поэтому возможны три случая:

1) $\pi_p < -\varepsilon$, где ε — абсолютный допуск на значение множителя строки. В данном случае строка p должна быть активным ограничением на своей нижней границе L_p, чтобы обеспечить двойственную допустимость решения исходной задачи. Если при этом верхняя граница строки отсутствует ($L_p = -\infty$), то задача не имеет двойственных допустимых решений;

2) $\pi_p > +\varepsilon$. В данном случае строка p должна быть активным ограничением на своей верхней границе U_p, и если эта граница отсутствует ($U_p = +\infty$), то задача не имеет двойственных допустимых решений;

3) $-\varepsilon \leq \pi_p \leq +\varepsilon$. В данном случае активной границей строки p может быть как ее верхняя, так и нижняя граница, поскольку это не влияет на двойственную допустимость решения.

Таким образом, во всех трех случаях исходное ограничение-неравенство (2) можно заменить ограничением-равенством, правой частью которого является нижняя или верхняя граница строки p, а также снять границы столбца q, в результате чего он станет свободным. Заметим, что после данного преобразования к столбцу q можно применить преобразование, рассмотренное в подразд. «Обработка столбцового синглета (неявная переменная недостатка)».

Восстановление базисного решения

Статус строки p в решении исходной задачи определяется ее статусом в решении преобразованной задачи и признаком границы, которая была выбрана в качестве активной:

Статус строки p (преобразованная задача)	Активная граница	Статус строки p (исходная задача)
GLP_BS	L_p	GLP_BS
GLP_BS	U_p	GLP_BS
GLP_NS	L_p	GLP_NL
GLP_NS	U_p	GLP_NU

Значение множителя строки π_p (как, впрочем, и значение столбца q) в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Восстановление решения внутренней точки

Значение множителя строки π_p в решении исходной задачи совпадает с его значением в решении преобразованной задачи.

Восстановление целочисленного решения

Не требуется.

2.10 Обработка строчного дублета (равенство)

Спецификация

```
#include "glpnpp.h"
NPPCOL *npp_eq_doublet(NPP *npp, NPPROW *p);
```

Назначение

Подпрограмма `npp_eq_doublet` обрабатывает строку p, которая является ограничением-равенством и имеет ровно два ненулевых коэффициента:

$$a_{pq}x_q + a_{pr}x_r = b. \tag{1}$$

В результате обработки один из столбцов q или r исключается из всех остальных строк ограничений и становится, таким образом, столбцовым синглетом типа «неявная переменная недостатка». При этом строка p не изменяется и вместе со столбцами q и r остается в задаче.

Возвращаемое значение

Подпрограмма `npp_eq_doublet` возвращает указатель на описатель того столбца q или r, который был исключен из остальных строк ограничений. Если по каким-либо причинам исключение не было выполнено, подпрограмма возвращает `NULL`.

Преобразование задачи

Вначале необходимо определить, какой из столбцов q или r следует исключать. Допустим, что таким столбцом является столбец q, и рассмотрим строку i-го ограничения, в которую этот столбец входит с ненулевым коэффициентом $a_{iq} \neq 0$:

$$L_i \leq \sum_j a_{ij}x_j \leq U_i. \tag{2}$$

Чтобы исключить столбец q из строки (2) достаточно вычесть из этой строки строку (1), умноженную на $\gamma_i = a_{iq}/a_{pq}$, т. е. заменить в преобразованной задаче строку (2) указанной линейной комбинацией этой строки и строки (1). Такое преобразование изменяет только коэффициенты столбцов q и r, а также границы строки i, новые значения которых будут равны:

$$\overline{a}_{iq} = a_{iq} - \gamma_i a_{pq} = 0, \tag{3}$$

$$\overline{a}_{ir} = a_{ir} - \gamma_i a_{pr}, \tag{4}$$

$$\overline{L}_i = L_i - \gamma_i b, \tag{5}$$

$$\overline{U}_i = U_i - \gamma_i b. \tag{6}$$

Восстановление базисного решения

Рассмотренное преобразование прямой системы исходной задачи:

$$L \le Ax \le U \tag{7}$$

эквивалентно умножению слева преобразующей матрицы F на компоненты этой системы, которая в преобразованной задаче принимает следующий вид:

$$FL \le FAx \le FU \quad \Rightarrow \quad \overline{L} \le \overline{A}x \le \overline{U}. \tag{8}$$

При этом матрица F имеет следующую структуру:

$$F = \begin{pmatrix} 1 & & & -\gamma_1 & & \\ & 1 & & -\gamma_2 & & \\ & & \ddots & \vdots & & \\ & & 1 & -\gamma_{p-1} & & \\ & & & 1 & & \\ & & & -\gamma_{p+1} & 1 & \\ & & & \vdots & & \ddots \end{pmatrix}, \tag{9}$$

где столбец, содержащий элементы $-\gamma_i$, соответствует строке p прямой системы.

Из (8) следует, что в результате указанного преобразования двойственная система исходной задачи:

$$A^T\pi + \lambda = c, \tag{10}$$

в преобразованной задаче будет следующей:

$$A^T F^T F^{-T}\pi + \lambda = c \quad \Rightarrow \quad \overline{A}^T\overline{\pi} + \lambda = c, \tag{11}$$

где

$$\overline{\pi} = F^{-T}\pi \tag{12}$$

есть вектор множителей строк в преобразованной задаче. Поэтому:

$$\pi = F^T\overline{\pi}. \tag{13}$$

Таким образом, как показывает соотношение (13), значение множителя строки p в решении исходной задачи можно вычислить следующим образом:

$$\pi_p = \overline{\pi}_p - \sum_i \gamma_i\overline{\pi}_i, \tag{14}$$

где $\overline{\pi}_i = \pi_i$ — множитель строки i $(i \ne p)$.

Заметим, что статусы всех строк и столбцов при переходе от решения преобразованной задачи к решению исходной задачи не изменяются.

Восстановление решения внутренней точки

Множитель строки p в решении исходной задачи вычисляется по формуле (14).

Восстановление целочисленного решения

Не требуется.

2.11 Обработка форсирующей строки

Спецификация

```
#include "glpnpp.h"
int npp_forcing_row(NPP *npp, NPPROW *p, int at);
```

Назначение

Подпрограмма `npp_forcing_row` обрабатывает строку p общего вида:

$$L_p \le \sum_j a_{pj} x_j \le U_p, \tag{1}$$

$$l_j \le x_j \le u_j, \tag{2}$$

где $L_p \le U_p$ и $l_j < u_j$ для всех $a_{pj} \ne 0$. При этом для данной строки известно, что:

1) если $at = 0$, то $|L_p - U_p^*| \le \varepsilon$, где U_p^* — неявная нижняя граница этой строки (см. ниже), ε — абсолютный допуск на значение строки;

2) если $at = 1$, то $|U_p - L_p^*| \le \varepsilon$, где L_p^* — неявная верхняя граница этой строки (см. ниже).

Возвращаемое значение

0 — успешная обработка;
1 — некоторые коэффициенты строки слишком малы.

Преобразование задачи

Неявные нижняя и верхняя границы строки (1) определяются соответствующими границами входящих в нее столбцов (переменных) следующим образом:

$$L_p^* = \inf \sum_j a_{pj} x_j = \sum_{j \in J^+} a_{pj} l_j + \sum_{j \in J^-} a_{pj} u_j, \tag{3}$$

$$U_p^* = \sup \sum_j a_{pj} x_j = \sum_{j \in J^+} a_{pj} u_j + \sum_{j \in J^-} a_{pj} l_j, \tag{4}$$

$$J^+ = \{j : a_{pj} > 0\}, \quad J^- = \{j : a_{pj} < 0\}. \tag{5}$$

Если $L_p \approx U_p^*$ $(at = 0)$, то (прямое) решение задачи может быть допустимым лишь когда переменные, входящие в строку (1), принимают свои граничные значения в соответствии с (4):

$$x_j = \begin{cases} u_j, & \text{если } j \in J^+ \\ l_j, & \text{если } j \in J^- \end{cases} \tag{6}$$

Аналогично, если $U_p \approx L_p^*$ ($at = 1$), то (прямое) решение задачи может быть допустимым лишь когда соответствующие переменные принимают свои граничные значения в соответствии с (3):

$$x_j = \begin{cases} l_j, & \text{если } j \in J^+ \\ u_j, & \text{если } j \in J^- \end{cases} \tag{7}$$

Условие (6) или (7) позволяет зафиксировать все столбцы (переменные), входящие в строку (1), на их границах и затем исключить их из задачи (см. подразд. «Обработка фиксированного столбца»). Так как при этом строка (1) становится избыточной, ее можно заменить *эквивалентной* свободной (неограниченной по знаку) строкой и также исключить из задачи (см. подразд. «Обработка свободной строки»).

Примечания:

1. Для применения данного преобразования необходимо, чтобы в строке (1) не было коэффициентов, абсолютная величина которых слишком мала, т. е. для всех a_{pj} необходимо выполнение условия:

$$|a_{pj}| \geq \varepsilon \cdot \max_k(1, |a_{pk}|), \tag{8}$$

где ε — относительный допуск на абсолютную величину коэффициентов строки. В противном случае фиксация столбцов является численно ненадежной и может привести к неверному решению задачи.

2. Подпрограмма фиксирует столбцы и снимает границы строки p, но оставляет эти строку и столбцы в преобразованной задаче.

Восстановление базисного решения

В преобразованной задаче строка p является неактивным ограничением и поэтому имеет статус `GLP_BS` (как результат последующего преобразования «Свободная строка»), а все столбцы, входящие в эту строку, являются небазисными и имеют статус `GLP_NS` (как результат последующего преобразования «Фиксированный столбец»).

Заметим, что в двойственной системе как преобразованной, так и исходной задачи каждому столбцу j, входящему в строку p соответствует следующая строка:

$$\sum_{i \neq p} a_{ij}\pi_i + a_{pj}\pi_p + \lambda_j = c_j, \tag{9}$$

откуда:

$$\lambda_j = c_j - \sum_{i \neq p} a_{ij}\pi_i - a_{pj}\pi_p. \tag{10}$$

В преобразованной задаче значения множителей всех строк π_i известны (включая множитель π_p, значение которого равен нулю, так как строка p неактивна). Поэтому, используя формулу (10), можно вычислить значения множителей λ_j для всех столбцов, входящих в строку p.

Заметим далее, что в исходной задаче столбцы, входящие в строку p, являются не фиксированными, а ограниченными. Поэтому статус GLP_NS каждого такого столбца следует заменить на GLP_NL или GLP_NU в зависимости от того на какой границе, нижней или верхней, был зафиксирован соответствующий столбец. Такое изменение статуса столбцов может привести к нарушению двойственной допустимости решения исходной задачи, так как теперь множители для столбцов должны удовлетворять условию:

$$\lambda_j \begin{cases} \geq 0, & \text{если статус столбца GLP\_NL,} \\ \leq 0, & \text{если статус столбца GLP\_NU.} \end{cases} \tag{11}$$

Если указанное условие выполняется, то решение исходной задачи совпадает с решением преобразованной задачи. В противном случае следует выполнить один вырожденный шаг прямого симплекс-метода для получения двойственно-допустимого (а значит, оптимального) решения исходной задачи следующим образом. Если при преобразовании задачи активной границей строки p была выбрана ее нижняя граница (случай $at = 0$), то мы меняем статус строки на GLP_NL (или GLP_NS) и начинаем *увеличивать* ее множитель ($\Delta\pi_p > 0$). Если же активной границей этой строки была выбрана ее верхняя граница (случай $at = 1$), то мы меняем статус строки на GLP_NU (или GLP_NS) и начинаем *уменьшать* ее множитель ($\Delta\pi_p < 0$). Из (10) следует, что:

$$\Delta\lambda_j = -a_{pj}\Delta\pi_p = -a_{pj}\pi_p. \tag{12}$$

При этом простой анализ формул (3)—(5) показывает, что при указанном изменении величины π_p множитель каждого столбца, имеющего статус GLP_NL (т. е. зафиксированного в преобразованной задаче на своей нижней границе) начнет *возрастать* ($\Delta\lambda_j > 0$), а множитель каждого столбца, имеющего статус GLP_NU (т. е. зафиксированного в преобразованной задаче на своей верхней границе) — *убывать* ($\Delta\lambda_j < 0$). Понятно, что в тот момент, когда *последний* множитель λ_q, для которого нарушено условие (11), обратится в нуль, множители всех столбцов будут иметь правильные знаки. Чтобы определить этот столбец допустим, что d_j — есть начальное значение множителя λ_j (т. е. относительная оценка столбца j) в преобразованной задаче, вычисленное по формуле (10) при $\pi_p = 0$. Тогда $\lambda_j = d_j + \Delta\lambda_j$, и из формулы (12) следует, что λ_j становится равным нулю, если:

$$\Delta\lambda_j = -a_{pj}\pi_p = -d_j \quad \Rightarrow \quad \pi_p = d_j/a_{pj}. \tag{13}$$

Следовательно, последним обратится в нуль множитель столбца q, для которого:

$$|d_q/a_{pq}| = \max_{j\in D} |\pi_p| = \max_{j\in D} |d_j/a_{pj}|, \tag{14}$$

где D — множество столбцов j, относительные оценки которых d_j имеют в преобразованной задаче неправильные знаки, т. е. не удовлетворяют

условию (11). (Таким образом, если $D = \varnothing$, то решение исходной задачи совпадает с решением преобразованной задачи и поэтому коррекция решения не требуется, как это было отмечено выше.) В решении исходной задачи столбец q получает статус `GLP_BS`, поскольку он заменяет в базисе столбец вспомогательной переменной строки p, которая теперь становится активным ограничением. При этом новое значение множителя строки p будет равно $\pi_p = d_q/a_{pq}$. Отметим, что вследствие вырожденности решения значения всех столбцов, входящих в строку p, при переходе от решения преобразованной задачи к решению исходной задачи не изменяются.

Восстановление решения внутренней точки

Коррекция значения множителя строки π_p в решении исходной задачи выполняется (если это необходимо) так же, как и в случае базисного решения. Значения всех столбцов, входящих в строку p, в решении исходной задачи совпадают с их значениями в решении преобразованной задачи.

Восстановление целочисленного решения

Не требуется.

2.12 Анализ строки общего вида

Спецификация

```
#include "glpnpp.h"
int npp_analyze_row(NPP *npp, NPPROW *p);
```

Назначение

Подпрограмма `npp_analyze_row` выполняет анализ строки (ограничения) p общего вида:

$$L_p \le \sum_j a_{pj} x_j \le U_p, \tag{1}$$

$$l_j \le x_j \le u_j, \tag{2}$$

где $L_p \le U_p$ и $l_j \le u_j$ для всех $a_{pj} \ne 0$.

Возвращаемое значение

0x?0 — нижняя граница строки отсутствует или является избыточной;
0x?1 — нижняя граница строки может быть активной;
0x?2 — нижняя граница строки является форсирующей;
0x0? — верхняя граница строки отсутствует или является избыточной;
0x1? — верхняя граница строки может быть активной;
0x2? — верхняя граница строки является форсирующей;
0x33 — границы строки противоречат границам столбцов.

Алгоритм

Анализ строки (1) основан на анализе ее неявных нижней и верхней границы, которые определяются соответствующими границами входящих в нее столбцов (переменных):

$$L_p^* = \inf \sum_j a_{pj} x_j = \sum_{j \in J^+} a_{pj} l_j + \sum_{j \in J^-} a_{pj} u_j, \tag{3}$$

$$U_p^* = \sup \sum_j a_{pj} x_j = \sum_{j \in J^+} a_{pj} u_j + \sum_{j \in J^-} a_{pj} l_j, \tag{4}$$

$$J^+ = \{j : a_{pj} > 0\}, \quad J^- = \{j : a_{pj} < 0\}. \tag{5}$$

(Заметим, что границы всех столбцов, входящих в строку p, предполагаются корректными, т. е. $l_j \le u_j$, поэтому $L_p^* \le U_p^*$.)

Анализ нижней границы строки L_p включает следующие случаи:

1) если $L_p > U_p^* + \varepsilon$, где ε — абсолютный допуск на значение строки, то нижняя граница строки L_p противоречит ее неявной верхней границе U_p^*, а значит, задача не имеет (прямых) допустимых решений;

2) если $U_p^* - \varepsilon \leq L_p \leq U_p^* + \varepsilon$, т. е. если $L_p \approx U_p^*$, то строка является форсирующей на своей нижней границе (см. разд. «Обработка форсирующей строки»);

3) если $L_p > L_p^* + \varepsilon$, то нижняя граница строки L_p может быть активной (без учета других строк, входящих в задачу);

4) если $L_p \leq L_p^* + \varepsilon$, то нижняя граница строки L_p не может быть активной, а значит, является избыточной и может быть исключена.

Анализ верхней границы строки U_p выполняется аналогично и включает следующие случаи:

1) если $U_p < L_p^* - \varepsilon$, то верхняя граница строки U_p противоречит ее неявной нижней границе L_p^*, а значит, задача не имеет (прямых) допустимых решений;

2) если $L_p^* - \varepsilon \leq U_p \leq L_p^* + \varepsilon$, т. е. если $U_p \approx L_p^*$, то строка является форсирующей на своей верхней границе (см. разд. «Обработка форсирующей строки»);

3) если $U_p < U_p^* - \varepsilon$, то верхняя граница строки U_p может быть активной (без учета других строк, входящих в задачу);

4) если $U_p \geq U_p^* - \varepsilon$, то верхняя граница строки U_p может быть активной, а значит, является избыточной и может быть исключена.

2.13 Удаление избыточной границы строки

Спецификация

```
#include "glpnpp.h"
void npp_inactive_bound(NPP *npp, NPPROW *p, int which);
```

Назначение

Подпрограмма `npp_inactive_bound` удаляет нижнюю (если $which = 0$) или верхнюю (если $which = 1$) границу строки p:

$$L_p \leq \sum_j a_{pj} x_j \leq U_p,$$

которая (граница) предполагается избыточной.

Преобразование задачи

Если $which = 0$, текущая нижняя граница L_p строки p полагается равной $-\infty$. Если $which = 1$, текущая верхняя граница U_p строки p полагается равной $+\infty$.

Восстановление базисного решения

Если в решении преобразованной задачи строка p не является активным ограничением (`GLP_BS`), то в решении исходной задачи ее статус не изменяется. В противном случае статус строки p в решении исходной задачи определяется ее типом до выполнения преобразования и статусом в решении преобразованной задачи:

Тип строки	Параметр *which*	Статус строки в преобразованной задаче	Статус строки в исходной задаче
$\Sigma \geq L_p$	0	`GLP_NF`	`GLP_NL`
$\Sigma \leq U_p$	1	`GLP_NF`	`GLP_NU`
$L_p \leq \Sigma \leq U_p$	0	`GLP_NU`	`GLP_NU`
$L_p \leq \Sigma \leq U_p$	1	`GLP_NL`	`GLP_NL`
$\Sigma = L_p = U_p$	0	`GLP_NU`	`GLP_NS`
$\Sigma = L_p = U_p$	1	`GLP_NL`	`GLP_NS`

Восстановление решения внутренней точки

Не требуется.

Восстановление целочисленного решения

Не требуется.

2.14 Определение неявных границ столбцов

Спецификация

```
#include "glpnpp.h"
void npp_implied_bounds(NPP *npp, NPPROW *p);
```

Назначение

Подпрограмма `npp_implied_bounds` использует строку (ограничение) p общего вида:

$$L_p \le \sum_j a_{pj} x_j \le U_p, \tag{1}$$

$$l_j \le x_j \le u_j, \tag{2}$$

где $L_p \le U_p$ и $l_j \le u_j$ для всех $a_{pj} \ne 0$, чтобы определить *неявные* границы столбцов (переменных x_j), входящих в эту строку.

Подпрограмма сохраняет неявные границы столбцов l_j^* и u_j^* в дескрипторах этих столбцов (`NPPCOL`), при этом текущие границы столбцов l_j и u_j не изменяются. (Далее неявные границы столбцов можно использовать для усиления текущих границ; см. подразд. «Обработка неявной нижней границы столбца» и «Обработка неявной верхней границы столбца».)

Алгоритм

Текущие границы столбцов (2) определяют неявные нижнюю и верхнюю границы строки (1):

$$L_p^* = \inf \sum_j a_{pj} x_j = \sum_{j \in J^+} a_{pj} l_j + \sum_{j \in J^-} a_{pj} u_j, \tag{3}$$

$$U_p^* = \sup \sum_j a_{pj} x_j = \sum_{j \in J^+} a_{pj} u_j + \sum_{j \in J^-} a_{pj} l_j, \tag{4}$$

$$J^+ = \{j : a_{pj} > 0\}, \quad J^- = \{j : a_{pj} < 0\}. \tag{5}$$

(Заметим, что границы всех столбцов, входящих в строку p, предполагаются корректными, т. е. $l_j \le u_j$, поэтому $L_p^* \le U_p^*$.)

Если $L_p > L_p^*$ и (или) $U_p < U_p^*$, то нижняя и (или) верхняя граница строки (1) может быть активной, а значит, такая строка определяет некоторые неявные границы входящих в нее переменных.

Пусть x_k — некоторая переменная, входящая в ограничение (1) с ненулевым коэффициентом $a_{pk} \ne 0$. Рассмотрим вначале случай, когда

активной может быть нижняя граница строки ($L_p > L_p^*$):

$$\begin{aligned} &\sum_j a_{pj}x_j \geq L_p \quad \Rightarrow \quad \sum_{j\neq k} a_{pj}x_j + a_{pk}x_k \geq L_p \quad \Rightarrow \\ &a_{pk}x_k \geq L_p - \sum_{j\neq k} a_{pj}x_j \quad \Rightarrow \quad a_{pk}x_k \geq L_p^k, \end{aligned} \tag{6}$$

где:

$$\begin{aligned} L_p^k &= \inf(L_p - \sum_{j\neq k} a_{pj}x_j) = L_p - \sup\sum_{j\neq k} a_{pj}x_j = \\ &= L_p - \sum_{j\in J^+\setminus\{k\}} a_{pj}u_j - \sum_{j\in J^-\setminus\{k\}} a_{pj}l_j. \end{aligned} \tag{7}$$

Таким образом:

$$x_k \geq l_k^* = L_p^k/a_{pk}, \text{ если } a_{pk} > 0; \tag{8}$$

$$x_k \leq u_k^* = L_p^k/a_{pk}, \text{ если } a_{pk} < 0; \tag{9}$$

где l_k^* и u_k^* — неявные нижняя и верхняя границы переменной x_k.

Рассмотрим теперь аналогичный случай, когда активной может быть верхняя граница строки ($U_p < U_p^*$):

$$\begin{aligned} &\sum_j a_{pj}x_j \leq U_p \quad \Rightarrow \quad \sum_{j\neq k} a_{pj}x_j + a_{pk}x_k \leq U_p \quad \Rightarrow \\ &a_{pk}x_k \leq U_p - \sum_{j\neq k} a_{pj}x_j \quad \Rightarrow \quad a_{pk}x_k \leq U_p^k, \end{aligned} \tag{10}$$

где:

$$\begin{aligned} U_p^k &= \sup(U_p - \sum_{j\neq k} a_{pj}x_j) = U_p - \inf\sum_{j\neq k} a_{pj}x_j = \\ &= U_p - \sum_{j\in J^+\setminus\{k\}} a_{pj}l_j - \sum_{j\in J^-\setminus\{k\}} a_{pj}u_j. \end{aligned} \tag{11}$$

Следовательно:

$$x_k \leq u_k^* = U_p^k/a_{pk}, \text{ если } a_{pk} > 0; \tag{12}$$

$$x_k \geq l_k^* = U_p^k/a_{pk}, \text{ если } a_{pk} < 0. \tag{13}$$

Примечание. В формулах (8), (9), (12) и (13) коэффициент a_{pk} не должен быть слишком малым по абсолютной величине относительно других ненулевых коэффициентов строки (1), т. е. необходимо выполнение условия:

$$|a_{pk}| \geq \varepsilon \cdot \max_j(1, |a_{pj}|), \tag{14}$$

где ε — относительный допуск на абсолютную величину коэффициентов строки. В противном случае неявные границы столбцов могут оказаться численно ненадежными. Например, использование формулы (8) для следующего ограничения-неравенства:

$$10^{-12}x_1 - x_2 - x_3 \geq 0,$$

где $x_1 \geq -1$, $x_2,\ x_3 \geq 0$, может привести к численно ненадежному заключению, что $x_1 \geq 0$.

Непосредственное использование формул (8), (9), (12) и (13) для вычисления неявных границ одной переменной требует выполнения $|J|$ операций, где $J = \{j : a_{pj} \neq 0\}$, что связано с необходимостью вычисления величин L_p^k и U_p^k. Поэтому вычисление неявных границ всех переменных, входящих в строку (1), потребовало бы выполнения $|J|^2$ операций, что является неудовлетворительным с практической точки зрения. Однако, общее число операций можно уменьшить до $|J|$ следующим образом.

Допустим, что $a_{pk} > 0$. Тогда из (7) и (11) имеем:

$$L_p^k = L_p - (U_p^* - a_{pk}u_k) = L_p - U_p^* + a_{pk}u_k,$$

$$U_p^k = U_p - (L_p^* - a_{pk}l_k) = U_p - L_p^* + a_{pk}l_k,$$

где L_p^* и U_p^* — неявные нижняя и верхняя границы строки, определяемые формулами (3) и (4). Подставляя полученные выражения в (8) и (12), получим:

$$l_k^* = L_p^k/a_{pk} = u_k + (L_p - U_p^*)/a_{pk}, \tag{15}$$

$$u_k^* = U_p^k/a_{pk} = l_k + (U_p - L_p^*)/a_{pk}. \tag{16}$$

Аналогично, если $a_{pk} < 0$, то в соответствии с (7) и (11) имеем:

$$L_p^k = L_p - (U_p^* - a_{pk}l_k) = L_p - U_p^* + a_{pk}l_k,$$

$$U_p^k = U_p - (L_p^* - a_{pk}u_k) = U_p - L_p^* + a_{pk}u_k.$$

Подставляя далее эти выражения в (8) и (12), получим:

$$l_k^* = U_p^k/a_{pk} = u_k + (U_p - L_p^*)/a_{pk}, \tag{17}$$

$$u_k^* = L_p^k/a_{pk} = l_k + (L_p - U_p^*)/a_{pk}. \tag{18}$$

Заметим, что формулы (15)—(18) применимы лишь в тех случаях, когда L_p^* и U_p^* конечны. Однако, если для какой-либо переменной x_j имеет место $l_j = -\infty$ и (или) $u_j = +\infty$, величины L_p^* (если $a_{pj} > 0$) и (или) U_p^* (если $a_{pj} < 0$) будут неопределенными. Рассмотрим поэтому наиболее общую ситуацию, когда некоторые границы переменных (2) могут отсутствовать.

Пусть:

$$J' = \{j : (a_{pj} > 0 \text{ и } l_j = -\infty) \text{ или } (a_{pj} < 0 \text{ и } u_j = +\infty)\}. \qquad (19)$$

Тогда (при условии, что верхняя граница строки U_p может быть активной) возможны три случая:

1) $J' = \varnothing$. В данном случае величина L_p^* конечна, а значит, для всех переменных x_j, входящих в строку (1), можно использовать формулы (16) и (17);

2) $J' = \{k\}$. В данном случае $L_p^* = -\infty$, но поскольку величина U_p^k (11) конечна, для переменной x_k можно использовать формулы (12) и (13). Заметим, что для всех остальных переменных x_j, $j \neq k$, имеет место $l_j^* = -\infty$ (если $a_{pj} < 0$) или $u_j^* = +\infty$ (если $a_{pj} > 0$);

3) $|J'| > 1$. В данном случае для всех переменных x_j, входящих в строку (1), имеет место $l_j^* = -\infty$ (если $a_{pj} < 0$) или $u_j^* = +\infty$ (если $a_{pj} > 0$), т. е. соответствующие границы столбцов не определены.

Аналогично, пусть:

$$J'' = \{j : (a_{pj} > 0 \text{ и } u_j = +\infty) \text{ или } (a_{pj} < 0 \text{ и } l_j = -\infty)\}. \qquad (20)$$

Тогда (при условии, что нижняя граница строки L_p может быть активной) также возможны три случая:

1) $J'' = \varnothing$. В данном случае величина U_p^* конечна, а значит, для всех переменных x_j, входящих в строку (1), можно использовать формулы (15) и (18);

2) $J'' = \{k\}$. В данном случае $U_p^* = +\infty$, но поскольку величина L_p^k (7) конечна, для переменной x_k можно использовать формулы (8) и (9). Заметим, что для всех остальных переменных x_j, $j \neq k$, имеет место $l_j^* = -\infty$ (если $a_{pj} > 0$) или $u_j^* = +\infty$ (если $a_{pj} < 0$);

3) $|J''| > 1$. В данном случае для всех переменных x_j, входящих в строку (1), имеет место $l_j^* = -\infty$ (если $a_{pj} > 0$) или $u_j^* = +\infty$ (если $a_{pj} < 0$), т. е. соответствующие границы столбцов не определены.

3 Группа целочисленных подпрограмм

3.1 Бинаризация задачи

Спецификация

```
#include "glpnpp.h"
int npp_binarize_prob(NPP *npp);
```

Назначение

Подпрограмма `npp_binarize_prob` заменяет в исходной задаче каждую целочисленную переменную:

$$l_q \le x_q \le u_q, \tag{1}$$

где $l_q < u_q$, эквивалентной суммой двоичных переменных.

Возвращаемое значение

Подпрограмма `npp_binarize_prob` возвращает число целочисленных переменных, для которых преобразование оказалось невозможным из-за того, что $u_q - l_q > d_{\max}$.

Преобразование задачи

Если переменная x_q имеет ненулевую нижнюю границу ($l_q \neq 0$), то вначале к ней применяется преобразование разд. «Обработка столбца с ненулевой нижней границей». Это позволяет считать, что:

$$0 \le x_q \le u_q. \tag{2}$$

Если $u_q = 1$, то переменная x_q уже является двоичной и дальнейшие преобразования не нужны. Допустим поэтому, что $2 \le u_q \le d_{\max}$, и определим наименьшее целое n такое, что $u_q \le 2^n - 1$ ($n \ge 2$, так как $u_q \ge 2$). В этом случае переменную x_q можно заменить суммой:

$$x_q = \sum_{k=0}^{n-1} 2^k x_k, \tag{3}$$

где x_k — двоичные переменные. Если при этом $u_q < 2^n - 1$, то в преобразованную задачу также необходимо включить дополнительное ограничение-неравенство:

$$\sum_{k=0}^{n-1} 2^k x_k \le u_q. \tag{4}$$

Примечание. Полагая, что в преобразованной задаче переменная x_q становится двоичной переменной x_0, данное преобразование приводит к появлению $n-1$ добавочных двоичных переменных.

Подставляя x_q из (3) в строку целевой функции, получим:

$$z = \sum_j c_j x_j + c_0 = \sum_{j \neq q} c_j x_j + c_q x_q + c_0 =$$
$$= \sum_{j \neq q} c_j x_j + c_q \sum_{k=0}^{n-1} 2^k x_k + c_0 = \sum_{j \neq q} c_j x_j + \sum_{k=0}^{n-1} c_k x_k + c_0,$$

где

$$c_k = 2^k c_q, \quad k = 0, \ldots, n-1. \tag{5}$$

Аналогично, подставляя x_q из (3) в строку i-го ограничения, получим:

$$L_i \leq \sum_j a_{ij} x_j \leq U_i \quad \Rightarrow \quad L_i \leq \sum_{j \neq q} a_{ij} x_j + a_{iq} x_q \leq U_i \quad \Rightarrow$$
$$L_i \leq \sum_{j \neq q} a_{ij} x_j + a_{iq} \sum_{k=0}^{n-1} 2^k x_k \leq U_i \quad \Rightarrow \quad L_i \leq \sum_{j \neq q} a_{ij} x_j + \sum_{k=0}^{n-1} a_{ik} x_k \leq U_i,$$

где

$$a_{ik} = 2^k a_{iq}, \quad k = 0, \ldots, n-1. \tag{6}$$

Восстановление решения

Значение переменной x_q вычисляется по формуле (3).

3.2 Проверка неравенства типа «упаковка»

Спецификация

```
#include "glpnpp.h"
int npp_is_packing(NPP *npp, NPPROW *row);
```

Возвращаемое значение

Если заданная строка (ограничение) является неравенством типа «упаковка» (см. ниже), подпрограмма `npp_is_packing` возвращает ненулевое значение. В противном случае подпрограмма возвращает нуль.

Неравенства типа «упаковка»

В каноническом формате неравенство типа «упаковка» (packing inequality) имеет следующий вид:

$$\sum_{j \in J} x_j \leq 1, \tag{1}$$

где все переменные x_j являются двоичными. Данное неравенство выражает условие, что в любом целочисленно-допустимом решении ненулевой (равной единице) может быть не более, чем одна переменная из множества J. Без ограничения общности можно считать, что $|J| \geq 2$, так как в случаях $J = \varnothing$ и $|J| = 1$ неравенство (1) является избыточным.

В общем случае неравенство типа «упаковка» может включать как исходные двоичные переменные x_j, так и их дополнения $\overline{x}_j$:

$$\sum_{j \in J^+} x_j + \sum_{j \in J^-} \overline{x}_j \leq 1, \tag{2}$$

где $J^+ \cap J^- = \varnothing$. Поэтому, используя подстановку $\overline{x}_j = 1 - x_j$, можно получить неравенство типа «упаковка» в обобщенном формате:

$$\sum_{j \in J^+} x_j - \sum_{j \in J^-} x_j \leq 1 - |J^-|. \tag{3}$$

3.3 Идентификация неравенства типа «упаковка»

Спецификация

```
#include "glpnpp.h"
int npp_hidden_packing(NPP *npp, NPPROW *row);
```

Назначение

Подпрограмма `npp_hidden_packing` обрабатывает заданное ограничение-неравенство, которое включает только двоичные переменные, причем число этих переменных должно быть не меньше двух. Если исходное ограничение эквивалентно неравенству типа «упаковка»,[8] подпрограмма заменяет это исходное ограничение указанным эквивалентным неравенством. При этом, если исходное ограничение-неравенство является двусторонним, то в случае необходимости оно заменяется парой односторонних ограничений.

Возвращаемое значение

Если исходное ограничение было заменено эквивалентным неравенством типа «упаковка», подпрограмма `npp_hidden_packing` возвращает ненулевое значение. В противном случае подпрограмма возвращает нуль.

Преобразование задачи

Рассмотрим ограничение-неравенство общего вида:[9]

$$\sum_{j \in J} a_j x_j \leq b, \tag{1}$$

где все переменные x_j являются двоичными и $|J| \geq 2$.

Пусть $J^+ = \{j : a_j > 0\}$, $J^- = \{j : a_j < 0\}$. Выполним переход к дополнительным переменным $x_j = 1 - \overline{x}_j$ для всех $j \in J^-$:

$$\sum_{j \in J} a_j x_j \leq b \quad \Leftrightarrow \quad \sum_{j \in J^+} a_j x_j + \sum_{j \in J^-} a_j x_j \leq b \quad \Leftrightarrow$$
$$\sum_{j \in J^+} a_j x_j + \sum_{j \in J^-} a_j (1 - \overline{x}_j) \leq b \quad \Leftrightarrow \quad \sum_{j \in J^+} a_j x_j - \sum_{j \in J^-} a_j \overline{x}_j \leq b - \sum_{j \in J^-} a_j.$$

С учетом указанного перехода будем считать, что в неравенстве (1) все коэффициенты являются положительными ($a_j > 0$). Кроме того,

[8] См. определение в подразд. «Проверка неравенства типа «упаковка».

[9] Неравенство вида '$\geq$' можно привести к указанному виду, умножая обе его части на -1.

без ограничения общности можно считать, что $a_j \leq b$. Действительно, допустим, что $a_j > b$. Тогда возможны три случая:

1) $b < 0$. В этом случае ограничение (1) является противоречивым, а значит, задача не имеет допустимых решений (см. разд. «Анализ строки общего вида»);

2) $b = 0$. В этом случае ограничение (1) является форсирующим на верхней границе (см. разд. «Обработка форсирующей строки»), откуда следует, что все переменные x_j должны быть зафиксированы в нуле;

3) $b > 0$. В этом случае ограничение (1) определяет неявную верхнюю границу переменной x_j равную нулю (см. разд. «Определение неявных границ столбцов»), что с учетом целочисленности приводит к фиксации этой переменной в нуле.

Предполагается, что все эти три случая будут распознаны во время базовой обработки задачи, предшествующей вызову подпрограммы `npp_hidden_packing`. Поэтому, если какой-либо из указанных случаев все же будет иметь место, обработку заданного ограничения-неравенства следует пропустить.

Итак, пусть $0 < a_j \leq b$ (с учетом перечисленных замечаний). Очевидно, что в этом случае ограничение-неравенство (1) эквивалентно неравенству типа «упаковка» если

$$a_j + a_k > b + \varepsilon \tag{2}$$

для всех $j, k \in J$, $j \neq k$, где ε — абсолютный допуск на значение строки (линейной формы). Проверка условия (2) для всех $j, k \in J$, $j \neq k$, требует времени $O(|J|^2)$. Однако, это время можно уменьшить до $O(|J|)$, если в качестве a_j и a_k выбрать два *наименьших* коэффициента — тогда, очевидно, условие (2) достаточно проверить один раз.

После замены исходного ограничения-неравенства (1) эквивалентным неравенством типа «упаковка» следует выполнить обратную подстановку $\overline{x}_j = 1 - x_j$ для всех $j \in J^-$ (см. выше).

Восстановление решения

Не требуется.

3.4 Идентификация релаксации типа «упаковка»

Спецификация

```
#include "glpnpp.h"
int npp_implied_packing(NPP *npp, NPPROW *row, int which,
      NPPCOL *var[], char set[]);
```

Назначение

Подпрограмма `npp_implied_packing` обрабатывает заданное ограничение (строку) общего вида:

$$L \leq \sum_j a_j x_j \leq U. \tag{1}$$

Если $which = 0$, то рассматривается только нижняя граница L, которая в этом случае должна существовать, а верхняя граница U игнорируется. Аналогично, если $which = 1$, то рассматривается только верхняя граница U, которая в этом случае должна существовать, а нижняя граница L игнорируется. Таким образом, если заданное ограничение является двусторонним неравенством или равенством, данную подпрограмму следует использовать дважды для каждой границы в отдельности.

Подпрограмма `npp_implied_packing` пытается определить нетривиальное (т. е. содержащее не менее двух двоичных переменных) неравенство типа «упаковка»:[10]

$$\sum_{j \in J^+} x_j - \sum_{j \in J^-} x_j \leq 1 - |J^-|, \tag{2}$$

которое является *релаксацией* ограничения (1) в том смысле, что всякое решение, удовлетворяющее этому ограничению также удовлетворяет неравенству (2). Если указанная релаксация существует, подпрограмма записывает указатели на описатели соответствующих двоичных переменных (столбцов) x_j и их признаки, соответственно, в элементы массивов $var[1]$, $var[2]$, ..., $var[len]$ и $set[1]$, $set[2]$, ..., $set[len]$, где $set[j] = 0$ означает, что $j \in J^+$, а $set[j] = 1$ означает что $j \in J^-$.

Возвращаемое значение

Подпрограмма `npp_implied_packing` возвращает значение len — общее число двоичных переменных в неравенстве (2), $len \geq 2$. Если же указанная релаксация не существует, то подпрограмма возвращает нуль.

[10] См. определение в подразд. «Проверка неравенства типа «упаковка».

Алгоритм

Если $which = 0$, умножим ограничение (1) на -1 и положим $b = -L$; если же $which = 1$, то положим $b = +U$. В обоих случаях это позволяет привести заданное ограничение к следующему виду:[11]

$$\sum_j a_j x_j \le b. \tag{3}$$

Пусть J — множество двоичных переменных, K^+ — множество недвоичных (непрерывных или целочисленных) переменных с коэффициентом $a_j > 0$, K^- — множество недвоичных переменных с коэффициентом $a_j < 0$. Тогда неравенство (3) можно записать следующим образом:

$$\sum_{j\in J} a_j x_j \le b - \sum_{j\in K^+} a_j x_j - \sum_{j\in K^-} a_j x_j. \tag{4}$$

Чтобы избавиться от недвоичных переменных заменим неравенство (4) следующим более слабым неравенством (релаксацией):

$$\sum_{j\in J} a_j x_j \le \overline{b}, \tag{5}$$

где

$$\begin{aligned} \overline{b} &= \sup(b - \sum_{j\in K^+} a_j x_j - \sum_{j\in K^-} a_j x_j) = \\ &= b - \inf \sum_{j\in K^+} a_j x_j - \inf \sum_{j\in K^-} a_j x_j = \\ &= b - \sum_{j\in K^+} a_j l_j - \sum_{j\in K^-} a_j u_j. \end{aligned} \tag{6}$$

Если при этом нижняя l_j (если $j \in K^+$) или верхняя u_j (если $j \in K^-$) граница какой-либо недвоичной переменной x_j не существует, то формально $\overline{b} = +\infty$, и в этом случае дальнейший анализ прекращается.

Пусть $B^+ = \{j \in J : a_j > 0\}$, $B^- = \{j \in J : a_j < 0\}$. Чтобы в неравенстве (5) сделать все коэффициенты положительными, заменим все x_j из B^- дополнительными переменными, т. е. выполним подстановку $x_j = 1 - \overline{x}_j$ для всех $j \in B^-$, в результате чего получим:

$$\sum_{j\in B^+} a_j x_j - \sum_{j\in B^-} a_j \overline{x}_j \le \overline{b} - \sum_{j\in B^-} a_j. \tag{7}$$

Полученное неравенство является релаксацией заданного ограничения (1) и относится к типу «двоичный рюкзак» (binary knapsack inequality).

[11] Заметим, что поскольку одна из границ исходного ограничения не учитывается, мы уже имеем релаксацию.

Запишем его в стандартном виде:

$$\sum_{j\in J} \alpha_j z_j \le \beta, \tag{8}$$

где:

$$\alpha_j = \begin{cases} +a_j, & \text{если } j \in B^+, \\ -a_j, & \text{если } j \in B^-, \end{cases} \tag{9}$$

$$z_j = \begin{cases} x_j, & \text{если } j \in B^+, \\ 1 - x_j, & \text{если } j \in B^-, \end{cases} \tag{10}$$

$$\beta = \overline{b} - \sum_{j\in B^-} a_j. \tag{11}$$

В неравенстве (8) все коэффициенты положительны, поэтому искомая релаксация типа «упаковка» для этого неравенства имеет вид:

$$\sum_{j\in P} z_j \le 1. \tag{12}$$

Очевидно, что искомое множество $P \subseteq J$ должно удовлетворять следующему условию:

$$\alpha_j + \alpha_k > \beta + \varepsilon \ \text{ для всех } j,k \in P, j \ne k, \tag{13}$$

где ε — абсолютный допуск на значение строки (линейной формы). Поэтому в качестве такого множества можно взять $P = \{j : \alpha_j > (\beta+\varepsilon)/2\}$. Если при этом в неравенстве (8) имеются коэффициенты α_k, для которых $\alpha_k \le (\beta+\varepsilon)/2$, но которые тем не менее удовлетворяют неравенству (13) для всех $j \in P$, то *одну* соответствующую переменную z_k (например, имеющую максимальный коэффициент α_k) также можно включить в P, увеличив тем самым общее число двоичных переменных в неравенстве (12) на единицу.

После формирования множества P для неравенства (12) следует выполнить подстановку, обратную к (10), чтобы вернуться к исходным двоичным переменным x_j. В результате выполнения этой обратной подстановки искомая релаксация типа «упаковка» примет окончательный вид (2), где $J^+ = J \cap B^+$ и $J^- = J \cap B^-$.

3.5 Проверка неравенства типа «покрытие»

Спецификация

```
#include "glpnpp.h"
int npp_is_covering(NPP *npp, NPPROW *row);
```

Возвращаемое значение

Если заданная строка (ограничение) является неравенством типа «покрытие» (см. ниже), подпрограмма `npp_is_covering` возвращает ненулевое значение. В противном случае подпрограмма возвращает нуль.

Неравенства типа «покрытие»

В каноническом формате неравенство типа «покрытие» (covering inequality) имеет следующий вид:

$$\sum_{j \in J} x_j \geq 1, \tag{1}$$

где все переменные x_j являются двоичными. Данное неравенство выражает условие, что в любом целочисленно допустимом решении переменные из множества J не могут быть равны нулю одновременно, т. е. хотя бы одна переменная должна быть ненулевой (равной единице). Без ограничения общности можно считать, что $|J| \geq 2$, так как в случае $J = \varnothing$ неравенство (1) является противоречивым, а в случае $|J| = 1$ — форсирующим.

В общем случае неравенство типа «покрытие» может включать как исходные двоичные переменные x_j, так и их дополнения $\overline{x}_j$:

$$\sum_{j \in J^+} x_j + \sum_{j \in J^-} \overline{x}_j \geq 1, \tag{2}$$

где $J^+ \cap J^- = \varnothing$. Поэтому, используя подстановку $\overline{x}_j = 1 - x_j$, можно получить неравенство типа «покрытие» в обобщенном формате:

$$\sum_{j \in J^+} x_j - \sum_{j \in J^-} x_j \geq 1 - |J^-|. \tag{3}$$

(Можно заметить, что неравенство (3) отсекает недопустимые решения, где $x_j = 0$ для всех $j \in J^+$ и $x_j = 1$ для всех $j \in J^-$, от множества целочисленно-допустимых решений.)

Примечание. Если $|J| = 2$, то неравенство (3) эквивалентно неравенству типа «упаковка» (см. подразд. «Проверка неравенства типа «упаковка»).

3.6 Идентификация неравенства типа «покрытие»

Спецификация

```
#include "glpnpp.h"
int npp_hidden_covering(NPP *npp, NPPROW *row);
```

Назначение

Подпрограмма `npp_hidden_covering` обрабатывает заданное ограничение-неравенство, которое включает только двоичные переменные, причем число этих переменных должно быть не меньше трех. Если исходное ограничение эквивалентно неравенству типа «покрытие»,[12] подпрограмма заменяет это исходное ограничение указанным эквивалентным неравенством. При этом, если исходное ограничение-неравенство является двусторонним, то в случае необходимости оно заменяется парой односторонних ограничений.

Возвращаемое значение

Если исходное ограничение было заменено эквивалентным неравенством типа «покрытие», подпрограмма `npp_hidden_covering` возвращает ненулевое значение. В противном случае подпрограмма возвращает нуль.

Преобразование задачи

Рассмотрим ограничение-неравенство общего вида:[13]

$$\sum_{j \in J} a_j x_j \geq b, \tag{1}$$

где все переменные x_j являются двоичными и $|J| \geq 3$.

Пусть $J^+ = \{j : a_j > 0\}$, $J^- = \{j : a_j < 0\}$. Выполним переход к дополнительным переменным $x_j = 1 - \overline{x}_j$ для всех $j \in J^-$:

$$\sum_{j \in J} a_j x_j \geq b \quad \Leftrightarrow \quad \sum_{j \in J^+} a_j x_j + \sum_{j \in J^-} a_j x_j \geq b \quad \Leftrightarrow$$
$$\sum_{j \in J^+} a_j x_j + \sum_{j \in J^-} a_j (1 - \overline{x}_j) \geq b \quad \Leftrightarrow \quad \sum_{j \in J^+} a_j x_j - \sum_{j \in J^-} a_j \overline{x}_j \geq b - \sum_{j \in J^-} a_j.$$

С учетом указанного перехода будем считать, что в неравенстве (1) все коэффициенты являются положительными ($a_j > 0$). Кроме того, без ограничения общности можно считать, что $b > 0$, так как в противном

[12] См. определение в подразд. «Проверка неравенства типа «покрытие».

[13] Неравенство вида '$\leq$' можно привести к указанному виду, умножая обе его части на -1.

случае ограничение (1) будет избыточным (см. разд. «Анализ строки общего вида»). Очевидно, что в этом случае ограничение-неравенство (1) эквивалентно неравенству типа «покрытие» если

$$a_j \geq b \tag{2}$$

для всех $j \in J$.

После замены ограничения-неравенства (1) эквивалентным неравенством типа «покрытие» следует выполнить обратную подстановку $\overline{x}_j = 1 - x_j$ для всех $j \in J^-$ (см. выше).

Восстановление решения

Не требуется.

3.7 Проверка равенства типа «разбиение»

Спецификация

```
#include "glpnpp.h"
int npp_is_partitioning(NPP *npp, NPPROW *row);
```

Возвращаемое значение

Если заданная строка (ограничение) является равенством типа «разбиение» (см. ниже), подпрограмма `npp_is_partitioning` возвращает ненулевое значение. В противном случае подпрограмма возвращает нуль.

Равенства типа «разбиение»

В каноническом формате равенство типа «разбиение» (partitioning equality) имеет следующий вид:

$$\sum_{j \in J} x_j = 1, \tag{1}$$

где все переменные x_j являются двоичными. Данное равенство выражает условие, что в любом целочисленно допустимом решении ровно одна переменная из множества J должна быть ненулевой (равной единице), а остальные переменные должны быть равны нулю. Без ограничения общности можно считать, что $|J| \geq 2$, так как в случае $J = \varnothing$ равенство (1) является противоречивым, а в случае $|J| = 1$ — фиксирующим.

В общем случае равенство типа «разбиение» может включать как исходные двоичные переменные x_j, так и их дополнения $\overline{x}_j$:

$$\sum_{j \in J^+} x_j + \sum_{j \in J^-} \overline{x}_j = 1, \tag{2}$$

где $J^+ \cap J^- = \varnothing$. Поэтому, используя подстановку $\overline{x}_j = 1 - x_j$, можно получить равенство типа «разбиение» в обобщенном формате:

$$\sum_{j \in J^+} x_j - \sum_{j \in J^-} x_j = 1 - |J^-|. \tag{3}$$

3.8 Редукция коэффициентов ограничения-неравенства

Спецификация

```
#include "glpnpp.h"
int npp_reduce_ineq_coef(NPP *npp, NPPROW *row);
```

Назначение

Подпрограмма `npp_reduce_ineq_coef` обрабатывает заданное ограничение-неравенство и пытается заменить его эквивалентным ограничением, где абсолютная величина коэффициентов при двоичных переменных меньше, чем в исходном ограничении. Если ограничение-неравенство является двусторонним, то в случае необходимости оно заменяется парой односторонних ограничений.

Возвращаемое значение

Подпрограмма `npp_reduce_ineq_coef` возвращает общее число редуцированных коэффициентов.

Обоснование метода

Рассмотрим ограничение-неравенство общего вида:[14]

$$\sum_{j \in J} a_j x_j \geq b. \tag{1}$$

Пусть x_k — двоичная переменная (остальные переменные могут быть как целочисленными, так и непрерывными). Запишем ограничение (1) в виде:

$$a_k x_k + t_k \geq b, \tag{2}$$

где

$$t_k = \sum_{j \in J \setminus \{k\}} a_j x_j. \tag{3}$$

Поскольку x_k — двоичная переменная, ограничение (2) эквивалентно *дизъюнкции* следующих двух ограничений:

$$x_k = 0, \quad t_k \geq b \tag{4}$$

или

$$x_k = 1, \quad t_k \geq b - a_k. \tag{5}$$

Допустим также, что для частичной суммы t_k известна ее некоторая неявная нижняя граница $\inf t_k$.[15]

[14] Неравенство вида '$\leq$' можно привести к указанному виду, умножая обе его части на -1.

[15] Такую неявную границу можно вычислить, например, используя текущие границы соответствующих переменных.

Случай $a_k > 0$ (см. рис. 1). Пусть $\inf t_k < b$, так как иначе оба ограничения (4) и (5), а поэтому и ограничение (2) будут избыточными. Если при этом $\inf t_k > b - a_k$, то избыточным будет только ограничение (5), которое тогда можно заменить следующим избыточным, а значит, *эквивалентным* ограничением:

$$t_k \geq b - \overline{a}_k = \inf t_k, \tag{6}$$

где

$$\overline{a}_k = b - \inf t_k. \tag{7}$$

Таким образом, исходное ограничение (2) эквивалентно следующему ограничению с измененным коэффициентом при переменной x_k:

$$\overline{a}_k x_k + t_k \geq b. \tag{8}$$

При этом из $\inf t_k < b$ следует, что $\overline{a}_k > 0$, т. е. коэффициент при x_k сохраняет свой знак, а из $\inf t_k > b - a_k$ следует, что $\overline{a}_k < a_k$, т. е. происходит *редукция* абсолютной величины этого коэффициента.

Случай $a_k < 0$ (см. рис. 2). Пусть $\inf t_k < b - a_k$, так как иначе оба ограничения (4) и (5), а поэтому и ограничение (2) будут избыточными. Если при этом $\inf t_k > b$, то избыточным будет только ограничение (4), которое тогда можно заменить следующим эквивалентным избыточным ограничением:

$$t_k \geq \overline{b} = \inf t_k. \tag{9}$$

Записывая ограничение (5) в виде:

$$t_k \geq b - a_k = \overline{b} - \overline{a}_k, \tag{10}$$

где

$$\overline{a}_k = a_k + \overline{b} - b = a_k + \inf t_k - b, \tag{11}$$

получим, что дизъюнкция ограничений (9) и (10) эквивалентна дизъюнкции ограничений (4) и (5), а значит, исходное ограничение (2) эквивалентно следующему ограничению с измененными коэффициентом при переменной x_k и правой частью:

$$\overline{a}_k x_k + t_k \geq \overline{b}. \tag{12}$$

При этом из $\inf t_k < b - a_k$ следует, что $\overline{a}_k < 0$, т. е. коэффициент при x_k сохраняет свой знак, а из $\inf t_k > b$ следует, что $\overline{a}_k > a_k$, т. е. происходит *редукция* абсолютной величины этого коэффициента.

Таким образом, в обоих случаях при выполнении указанных условий ограничение (2) можно заменить эквивалентным ограничением с редуцированным коэффициентом $\overline{a}_k$ и (в случае $a_k < 0$) увеличенной правой частью $\overline{b}$.

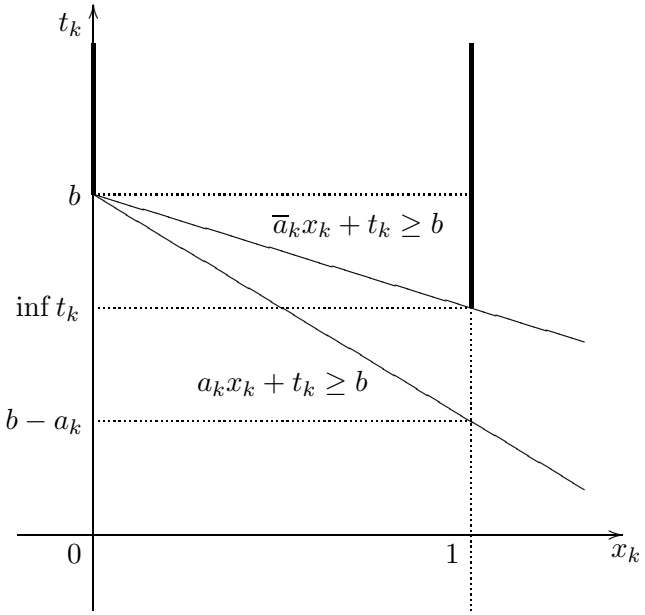

Рис. 1. Случай $a_k > 0$.

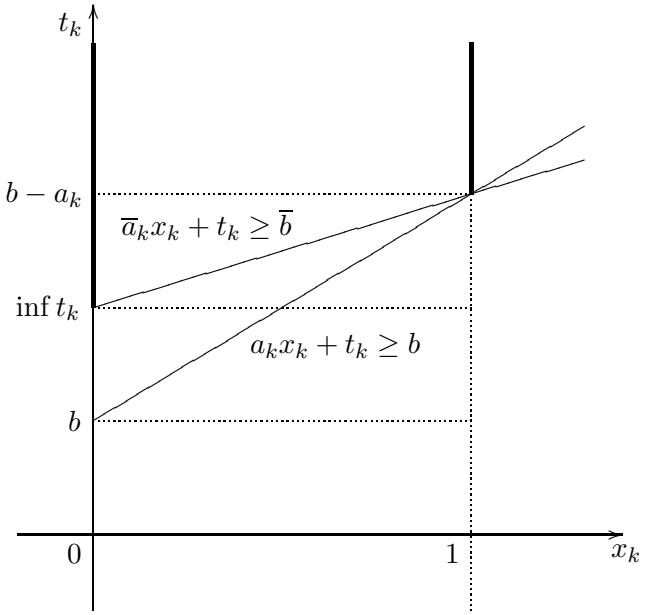

Рис. 2. Случай $a_k < 0$.

Преобразование задачи

В подпрограмме `npp_reduce_ineq_coef` используется следующая неявная нижняя граница частичной суммы (3):

$$\inf t_k = \sum_{j \in J^+ \setminus \{k\}} a_j l_j + \sum_{j \in J^- \setminus \{k\}} a_j u_j, \tag{13}$$

где $J^+ = \{j : a_j > 0\}$, $J^- = \{j : a_j < 0\}$, l_j и u_j — нижняя и верхняя границы переменной x_j.

Чтобы сократить общее число операций, необходимых для вычисления $\inf t_k$ для каждой двоичной переменной, вместо формулы (13) используется эквивалентная формула:

$$\inf t_k = \begin{cases} h - a_k l_k \ = h, & \text{если } a_k > 0, \\ h - a_k u_k = h - a_k, & \text{если } a_k < 0, \end{cases} \tag{14}$$

где

$$h = \sum_{j \in J^+} a_j l_j + \sum_{j \in J^-} a_j u_j \tag{15}$$

представляет собой неявную нижнюю границу линейной формы (1).

Редукция положительного коэффициента ($a_k > 0$) не изменяет значения h, поскольку $l_k = 0$. В случае редукции отрицательного коэффициента ($a_k < 0$) из (11) следует, что:

$$\Delta a_k = \overline{a}_k - a_k = \inf t_k - b \ (> 0), \tag{16}$$

поэтому новое значение h (учитывая, что $u_k = 1$) можно определить следующим образом:

$$h := h + \Delta a_k = h + (\inf t_k - b). \tag{17}$$

Восстановление решения

Не требуется.

Литература

[1] K. L. Hoffman, M. Padberg, "Improving LP-Representations of Zero-One Linear Programs for Branch-and-Cut." ORSA J. on Computing, Vol. 3, No. 2 (1991), pp. 121-34.

[2] R. Fourer and D. M. Gay, "Experience with a Primal Presolve Algorithm." Numerical Analysis Manuscript 93-06, AT&T Bell Laboratories, New Jersey, U.S.A. (1993).

[3] J. Gondzio, "Presolve Analysis of Linear Programs Prior to Applying an Interior Point Method." Tech. Rep. 1994.3, Logilab, University of Geneva, Geneva, Switzerland (1994).

[4] M. W. P. Savelsbergh, "Preprocessing and Probing Techniques for Mixed Integer Programming Problems." INFORMS J. on Comput., Vol. 6, Iss. 4 (Fall 1994), pp. 445-54.

[5] A. Świętanowski, "A Modular Presolve Procedure for Large Scale Linear Programming." WP-95-113, IIASA, Laxenburg, Austria (1995).