EESTI NSV TEADUSTE AKADEEMIA TOIMETISED.
FÜÜSIKA * MATEMAATIKA
ИЗВЕСТИЯ АКАДЕМИИ НАУК ЭСТОНСКОЙ ССР. ФИЗИКА * МАТЕМАТИКА
PROCEEDINGS OF THE ACADEMY OF SCIENCES OF THE ESTONIAN SSR.
PHYSICS * MATHEMATICS

1985, 34, 1

https://doi.org/10.3176/phys.math.1985.1.02

УДК 519.854.62

Л. КИВИСТИК

ОБ УСКОРЕНИИ ПОЛНОСТЬЮ ЦЕЛОЧИСЛЕННОГО АЛГОРИТМА ГОМОРИ

(Представил А. Хумал)

Для ускорения полностью целочисленного алгоритма Гомори предлагается использовать замещающие ограничения, коэффициенты которых получаются решением вспомогательных задач линейного программирования. Как показал эксперимент, проведенный на ЭВМ, такой способ часто весьма эффективен и значительно уменьшает нерегулярность алгоритма.

1. Постановка проблемы и построение вспомогательных задач. Рассмотрим полностью целочисленный алгоритм Гомори [1, 2] для решения задачи линейного целочисленного программирования: максимизировать функцию

$$x_0 = a_{00} + \sum_{j \in J} a_{0j}(-x_j) \tag{1}$$

при условиях

$$x_i = a_{i0} + \sum_{j \in J} a_{ij}(-x_j) \geqslant 0 \quad (i \in I), \tag{2}$$

$$x_j = -1\,(-x_j) \geqslant 0 \quad (j \in J), \tag{3}$$

$$x_j \text{ — целое} \quad (j = 0, 1, \ldots, N), \tag{4}$$

где I — множество базисных, J — множество небазисных переменных ($I \cup J = \{1, 2, \ldots, N\}$), а все коэффициенты — целые числа. Здесь и в дальнейшем пользуются обозначениями из [3], в частности, через A_j обозначается j-й столбец симплексной таблицы,

$$A_j = (a_{0j}, a_{1j}, \ldots, a_{Nj})^{\mathrm{T}},$$

причем предполагается, что симплексная таблица l-нормальна, т. е. $A_j > 0$ при всех $j \in J$.

Как известно, одни задачи целочисленного программирования разрешаются методами отсечения (в том числе и полностью целочисленным алгоритмом Гомори) через несколько итераций, для решения других задач той же размерности могут потребоваться тысячи и миллионы итераций. Такое нежелательное явление принято называть нерегулярностью этих методов. Ниже в этом смысле будем говорить также о нерегулярных (и регулярных) вариантах алгоритма. В [4] приведен пример двухпараметрического класса «плохих» задач, требующих в случае полностью целочисленного алгоритма Гомори (при любом выборе одного из возможных заданных ограничений в качестве производящего) по меньшей мере $(q^{r-2}+r-2)/r+1$ итерацию, где q и r — параметры этого класса (q — наибольший по абсолютной величине коэффициент в ограничениях и $r = |I|$). Например, уже при $q = r = 10$ число итераций огромное: оно не меньше, чем 10^7+2.

Поставим цель уменьшить число итераций алгоритма за счет того, что в качестве производящего ограничения выбирается некоторое подходящее замещающее ограничение

$$\sum_{i\in I} a_{i0}y_i + \sum_{j\in J}\left(\sum_{i\in I} a_{ij}y_i\right)(-x_j) \geqslant 0, \tag{5}$$

где $y_i \geqslant 0$. При этом коэффициенты y_i можно всегда определить так, чтобы в отсечении (см., напр., [1–3]) было $\lambda = 1$. Следовательно, искомое отсечение имеет вид

$$x_{N+r} = \left[\sum_{i\in I} a_{i0}y_i\right] + \sum_{j\in J}\left[\sum_{i\in I} a_{ij}y_i\right](-x_j) \geqslant 0, \tag{6}$$

где x_{N+r} — новое переменное. Вследствие симплексной итерации получим

$$A_0 := A_0 - \left(-\left[\sum_{i\in I} a_{i0}y_i\right]\right)A_l$$

(перед символом := стоит новое текущее значение вектора A_0), где ведущий столбец A_l определяется из условия

$$A_l = \operatorname{lex}\min\left\{A_j \middle| \sum_{i\in I} a_{ij}y_i < 0\right\}. \tag{7}$$

Итак, вследствие итерационного шага вектор A_0 уменьшается на вектор

$$-\left[\sum_{i\in I} a_{i0}y_i\right]A_l > 0. \tag{8}$$

Поэтому, чем больше (лексикографически) этот вектор, тем больше уменьшается A_0 и, вероятно, тем быстрее достигается оптимальное решение. В настоящей работе коэффициенты y_i определяются именно из условия лексикографической максимизации вектора (8). В случае, когда векторы A_j, которые могут выступать в качестве ведущего столбца A_l, сильно (лексикографически) отличаются друг от друга, в частности, имеют различные степени вырожденности [3], то вместо максимизации вектора (8) можно потребовать максимизации вектора A_l.

Так как вектор A_l следует определить из условия (7), то столбец A_l является ведущим, а его коэффициент в (8) максимальным, если а) $-1 \leqslant \sum_{i\in I} a_{il}y_i < 0$, б) $\sum_{i\in I} a_{ij}y_i \geqslant 0$ для всех лексикографически меньших столбцов A_j и в) величина $y_0 = \sum_{i\in I} a_{i0}y_i$ принимает минимальное отрицательное значение. Итак, поставленная цель достигается решением ряда задач линейного программирования и сравнением векторов (8) при разных значениях индекса l.

Чтобы облегчить выбор максимального вектора (8), поступаем следующим образом. Упорядочим все столбцы A_j, имеющие по крайней мере один отрицательный элемент в базисных строках, в порядке лексикографического возрастания

$$A_{l_1} < A_{l_2} < \ldots < A_{l_w}. \tag{9}$$

Среди них выделим те векторы $A_{l_q}, A_{l_{q+1}}, \ldots, A_{l_w}$ $(q \geqslant 1)$, степень вырожденности которых меньше k_0, где

$$k_0 = \min\{i \in I \mid a_{i0} < 0\} \tag{10}$$

(см. [3], формула (10)). Дальше, для каждого вектора A_{l_p} $(p = q, q+1, \ldots, w)$ найдем следующие за ним в (9) векторы $A_{l_{p+1}}, A_{l_{p+2}}, \ldots, A_{l_{a(p)}}$, имеющие ту же самую степень вырожденности, что и A_{l_p} (если такие

существуют), и определим наибольшие целые числа $\mu_{l_p j} \geqslant 1$, удовлетворяющие условиям (ср. [1])

$$\mu_{l_p j} A_{l_p} < A_j \quad (j = l_{p+1}, l_{p+2}, \ldots, l_{u(p)}).$$

Построим вспомогательные задачи линейного программирования ($p = q, q+1, \ldots, w$):

$$y_0 = \sum_{i \in I} a_{i0} y_i \to \min, \tag{11}$$

$$y_i \geqslant 0 \quad (i \in I), \tag{12}$$

$$\sum_{i \in I} a_{ij} y_i \geqslant 0 \quad (j = l_1, \ldots, l_{p-1}), \tag{13}$$

$$\sum_{i \in I} a_{il_p} y_i \geqslant -1, \tag{14}$$

$$\sum_{i \in I} a_{ij} y_i \geqslant -\mu_{l_p j} \quad (j = l_{p+1}, l_{p+2}, \ldots, l_{u(p)}) \tag{15}$$

(в случае, когда степень вырожденности вектора $A_{l_{p+1}}$ меньше степени вырожденности вектора A_{l_p}, или же $p = w$, ограничения (15) отсутствуют). Если существует оптимальное (или допустимое) решение $y_i{}^p$ ($i \in I$) задачи (11)—(15), при котором y_0 и левая часть неравенства (14) строго отрицательны, то компоненты этого оптимального (или допустимого) решения подходят в качестве коэффициентов y_i замещающего ограничения (5), определяющего ведущий столбец A_{l_p}. В соответствующем отсечении (6) $y_i = y_i{}^p$.

Чтобы выделить те задачи, у которых левая часть условия (14) не может при допустимых решениях принимать отрицательного значения, а в случае возможности обеспечить отрицательность левой части условия (14), можно заменить условие (14) на условие

$$0 > \sum a_{il_p} y_i \geqslant -1. \tag{16}$$

Однако мы этого не сделаем, так как этим усложняется практическое решение вспомогательных задач. Условие (16) проверим после решения задачи (11)—(15).

2. Свойства вспомогательных задач. С увеличением индекса p ограничения (14) и (15) заменяются более строгими (а иногда добавляются еще новые ограничения типа (14) и (15)). Поэтому из существования у задачи при некотором индексе $p = r$ только тривиального решения $y_i = 0$ ($i \in I$) следует существование у нее только тривиального решения также при всех $p > r$. В таком случае нас могут интересовать только задачи с индексами $p < r$. По той же причине из неограниченности целевой функции задачи (11)—(15) при некотором индексе $p = t$ следует неограниченность целевой функции (11) при каждом индексе $p < t$. В таком случае представляют интерес только те из задач (11)—(15), для которых $p > t$. В дальнейшем являются полезными еще следующие менее тривиальные свойства задач (11)—(15).

1. *Если некоторые из задач* (11)—(15) *соответствуют столбцам с одной и той же степенью вырожденности, то они либо все неразрешимы, либо все разрешимы.*

2. *Если при* $p = w$ *задача* (11)—(15) *неразрешима, то задача* (1)—(4) *тоже неразрешима.*

3. *Если при некотором индексе* $p = r$ *задача* (11)—(15) *неразрешима* (*т. е. ее целевая функция неограничена на множестве допустимых решений*), *то при* $p = r+1$ *задача имеет допустимое решение со значением целевой функции* $y_0 < 0$.

4. *Если задача* (1)—(4) *разрешима, то существует разрешимая задача* (11)—(15), *оптимальное решение* y_i^p $(i \in I)$ *которой удовлетворяет условиям*

$$y_0^p = \sum_{i \in I} a_{i0} y_i^p < 0, \tag{17}$$

$$-1 \leqslant \sum_{i \in I} a_{il_p} y_i^p < 0. \tag{18}$$

Доказательство свойств 1 и 2 можно провести, например, на основе следующего свойства двойственных задач ([5], с. 156, следствие 3.2): для того чтобы одна из двойственных задач имела допустимые решения, а множество допустимых решений другой задачи было пусто, необходима и достаточна неограниченность целевой функции первой задачи на множестве ее допустимых решений.

Возьмем в качестве первой одну из задач (11)—(15). Так как она имеет тривиальное допустимое решение $y_i = 0$ $(i \in I)$, то ее неразрешимость означает неограниченность целевой функции (11). Тогда двойственная ей задача не имеет допустимых решений. Но всем рассматриваемым в свойстве 1 задачам соответствуют двойственные задачи с одной и той же системой ограничений. Из этого и следует справедливость свойства 1.

Доказательство свойства 2 следует из того, что замена в последнем ограничении (14) (ограничения (15) теперь отсутствуют) правой части на 0 также не меняет неограниченности функции (11). Но тогда существует допустимое решение $y_i \geqslant 0$ $(i \in I)$ с $y_0 < 0$, при котором выполняются неравенства (13) для $j = l_1, \ldots, l_w$. Это значит, что из ограничений (2)—(4) следует противоречивое неравенство (6), где вместо y_i стоят компоненты этого допустимого решения. Следовательно, задача (1)—(4) неразрешима.

Доказательство свойства 3 достаточно провести для случая, когда степень вырожденности столбца $A_{l_{r+1}}$ меньше степени вырожденности столбца A_{l_r} (иначе свойство 3 следует из свойства 1). Тогда ограничения (15) отсутствуют. Заменим в (14), где $p = r$, правую часть на 0. Так как это не изменяет неограниченности целевой функции (11), то задача (11)—(13), где $j = l_1, \ldots, l_r$, имеет допустимое решение со значением целевой функции $y_0 = \sum a_{i0} y_i < 0$. Но вместе с каждым допустимым решением y_i $(i \in I)$ этой задачи является допустимым решением также εy_i $(i \in I)$ для любого $\varepsilon > 0$, причем y_0 остается отрицательным. Добавим новые ограничения (14), где $p = r+1$, и (15), где $j = l_{r+2}, \ldots, l_{r+k}$ (k — число столбцов A_j, имеющих с $A_{l_{r+1}}$ одинаковую степень вырожденности). Так как при $y_i = 0$ $(i \in I)$ эти новые ограничения удовлетворяются как строгие неравенства, то они остаются в силе при рассмотренном раньше решении εy_i $(i \in I)$, если только ε достаточно малое положительное число. Этим свойство 3 доказано.

Доказательство свойства 4. В силу свойств 2 и 1 из разрешимости задачи (1)—(4) следует разрешимость по крайней мере одной группы задач (11)—(15), соответствующих столбцам A_j одной и той же степени вырожденности. Рассмотрим оптимальное базисное решение y_i^r $(i \in I)$ некоторой задачи, при которой $y_0^r = \sum a_{i0} y_i^r < 0$. В случае существования неограниченных задач среди (11)—(15) такой задачей является по свойству 3 по крайней мере первая задача первой группы разрешимых задач. Если неограниченных задач вообще нет, то такой является, например, та из задач (11)—(15), которая соответствует некоторому столбцу A_{l_p}, определяемому основным или обобщенным правилом [3] выбора генерирующей строки (одно допустимое решение такой задачи имеет вид $y_k > 0$, $y_i = 0$, $i \neq k$, со значением целевой функции $y_0 = a_{k0} y_k < 0$; k — индекс генерирующей строки). Если при оптимальном базисном решении y_i^r $(i \in I)$ имеем $\sum a_{il_r} y_i^r < 0$, то оно и удовлетворяет условиям (17) и (18) при $p = r$. Если $\sum a_{il_r} y_i^r \geqslant 0$, то левая часть по крайней мере одного неравенства (15) при $y_i = y_i^r$ отрицательна. Действительно, иначе мы могли бы заменить во всех ограничениях (14)—(15) правые части нулями, не изменяя оптимального решения. Но единственной вершиной множества допустимых решений такой задачи и, следовательно, также единственным базисным решением ее является $y_i = 0$ $(i \in I)$, что противоречит неравенству $y_0^r < 0$.

Рассмотрим первое неравенство (15), в котором при $y_i=y_i^r$ левая часть отрицательна. Пусть это неравенство имеет индекс $j=l_s$. Тогда задача (11)—(15) имеет при $p=s$ допустимое решение

$$\bar{y}_i^s=y_i^r/\lambda \quad (\lambda \geqslant \max_{j=l_s,\ldots,l_{u(s)}} (\mu_{l_r j}/\mu_{l_s j}) \geqslant 1),$$

при котором $\bar{y}_0^s=\Sigma a_{i0}\bar{y}_i^s<0$ и $0>\Sigma a_{il_s}\bar{y}_i^s\geqslant -1$.

Если оптимальное базисное решение y_i^s $(i\in I)$ не удовлетворяет последнему условию, т. е. $\Sigma a_{il_s}y_i^s\geqslant 0$, то, как уже доказано, существует первый индекс $j=l_t$ $(t>s)$, при котором левая часть неравенства (15) отрицательна, а соответствующая система (11)—(15) при $p=t$ имеет допустимое решение $\bar{y}_i^t$ $(i\in I)$, удовлетворяющее условиям $\bar{y}_0^t<0$ и $0>\Sigma a_{il_t}\bar{y}_i^t\geqslant -1$. Продолжая таким образом (если нужно), приходим к индексу, при котором также оптимальное решение удовлетворяет условиям (17) и (18) (если это не происходит раньше, то оно имеет место при последней задаче рассматриваемой группы). Свойство 4 доказано.

3. Вычислительный алгоритм. Из свойств 1—4 следует, что по крайней мере приближенно можно реализовать поставленную цель — максимизировать вектор (8) — нахождением тех оптимальных решений задач (11)—(15), которые удовлетворяют условиям (17) и (18). Но возможно, что в случае, когда при оптимальном решении y_i^p $(i\in I)$ неравенство (18) не выполняется, все-таки существует допустимое решение $\bar{y}_i^p$ $(i\in I)$, удовлетворяющее условиям (17) и (18) и дающее при $l=l_p$ вектору (8) лексикографически большее значение, чем дает некоторое оптимальное решение, которое удовлетворяет (при большем индексе p) этим условиям. Поэтому, если непременно желаем найти максимальный вектор (8), мы должны в некоторых случаях решить дополнительные задачи. Эти дополнительные задачи строятся после решения задач (11)—(15) и имеют вместо ограничения (14) ограничение (16), которое вводится в виде

$$-\varepsilon \geqslant \sum_{i\in I} a_{il_p}y_i \geqslant -1, \tag{19}$$

где ε — близкое к нулю положительное число. При этом дополнительные задачи {(11)—(13), (19), (15)} достаточно строить только для разрешимых задач (11)—(15), имеющих наименьшую степень вырожденности и оптимальное решение которых не удовлетворяет условию (18) (но удовлетворяет условию (17)).

Кажется, что довольно хорошие результаты дает также вариант алгоритма, где дополнительные задачи с условием (19) совсем не используются. В дальнейшем рассматривается именно этот вариант алгоритма.

При практическом решении вспомогательных задач (11)—(15) можно учитывать, что они отличаются друг от друга только свободными членами. Поэтому их можно решить как одну задачу, симплексная таблица которой содержит несколько столбцов свободных членов.

Изложим наконец общую схему ускоренного полностью целочисленного алгоритма Гомори, пользующегося замещающими ограничениями. При этом предположим, что симплексная таблица целочисленна и находится уже в l-нормальной форме (в алгоритме не пользуются ограничениями (19)).

1. Проверим, найдется ли $a_{i0}<0$. Если нет, то оптимальное решение найдено.

2. Выделим те столбцы A_j, которые имеют по крайней мере один отрицательный элемент a_{ij} в строках с индексами $i\in I$, упорядочим эти столбцы в виде (9) и построим вспомогательные задачи (11)—(15) для индексов $p=q, q+1, \ldots, w$, где A_{l_q} — первый в (9) стол-

бец, степень вырожденности которого меньше, чем k_0 (k_0 определено формулой (10)).

3. Находим нетривиальные оптимальные решения для тех из задач (11)—(15), у которых такие решения существуют. Если их нет, то задача (1)—(4) неразрешима.

4. Выделим те оптимальные решения y_i^p ($i \in I$) задач (11)—(15), которые удовлетворяют условиям (17), (18).

5. Среди выделенных определим решение y_i^p ($i \in I$), а также столбец A_{l_p}, которые реализуют максимум вектора (8) (при $y_i = y_i^p$ и $l = l_p$).

6. Построим отсечение (6) с $y_i = y_i^p$, возьмем вектор его коэффициентов в качестве ведущей строки, а в нем ведущим — элемент —1 в столбце A_{l_p}. Проведем симплексную итерацию. Вернемся к пункту 1.

Изложенный алгоритм будем ниже называть ускоренным вариантом алгоритма Гомори, а основной вариант целочисленного алгоритма Гомори — базисным алгоритмом.

Конечность ускоренного варианта для разрешимой задачи (1)—(4) следует из теоремы, доказанной в [3]. Для задачи (1)—(4), которая может и не иметь допустимого решения, конечность можно гарантировать применением модификации алгоритма, аналогичной той, которая описывается в [2], с. 184—186.

Существуют задачи, при которых ускорение в результате применения изложенного алгоритма настолько велико, что целесообразность его использования не вызывает никаких сомнений. Например, нетрудно доказать, что упомянутые «плохие» задачи Ю. Ю. Финкельштейна [4] решаются ускоренным вариантом лишь через одну итерацию (независимо от значений параметров q и r). При этом требуется решить две вспомогательные задачи (11)—(15). О результатах вычислительного эксперимента при решении других задач пойдет речь ниже.

4. Результаты вычислительного эксперимента. Чтобы проверить эффективность ускоренного варианта алгоритма, автором настоящей статьи составлена программа для микро-ЭВМ «Apple II+». Эта программа позволяет решать задачи как базисным алгоритмом, так и ускоренным вариантом его. Кроме того, предусматривается возможность использования ускорения только на первых итерациях или же в тех случаях, когда значение целевой функции несколько раз подряд не изменяется. Этой же программой можно генерировать задачи с псевдослучайными целочисленными коэффициентами в зависимости от заданных параметров n, m, q и R. При этом n — число небазисных переменных, m — число ограничений, q определяет границы изменения коэффициентов задачи: от $-q$ до $q-1$ в ограничениях и от 1 до $2q$ в целевой функции; каждое отрицательное целое число R определяет свой набор псевдослучайных коэффициентов.

К настоящему времени программой генерировано и решено свыше 100 задач. Чтобы одна итерация не требовала слишком много времени, ограничивались небольшими значениями n. В таблице приведены результаты решения 100 задач, построенных следующим образом: после выбора R и q фиксировали n (в большинстве случаев брали $n=4$) и увеличивали m до тех пор, пока задача не становилась неразрешимой, затем брали наибольшее значение m, при котором задача оказывалась разрешимой. Кроме данных о применении базисного и ускоренного варианта алгоритма в таблицу включены данные о применении варианта алгоритма, где ускорением пользовались только на первой итерации. В таблице приведены данные для тех 30 задач, решение которых хотя бы одним вариантом алгоритма требовало более 40 итераций. Для остальных 70 задач приведены только крайние и

Задача					Базисный алгоритм		Ускорение на первой итерации			Ускоренный вариант алгоритма		
№	R	n	m	q	Число итераций	Время, с	Число итераций	Число вспом. итер.	Время, с	Число итераций	Число вспом. итер.	Время, с
1	—1	4	3	50	92	29	51	3	19	13	26	35
2	—3	6	9	10	4000+	1952+	11	7	21	5	17	42
3	—3	6	10	6	3500+	2481+	3500+	6	2455+	21	92	220
4	—7	4	5	50	227	82	4	5	7	2	9	10
5	—7	5	9	50	125	69	*254+	9	190+	12	30	70
6	—12	4	12	50	*2477+	800+	*645+	5	302+	12	33	72
7	—12	4	12	40	*569+	251+	*637+	5	335+	11	30	65
8	—12	4	12	30	2522	1148	356	5	194	12	42	81
9	—12	4	12	20	3482	2024	6	10	17	3	15	24
10	—12	4	12	10	659	346	209	5	127	14	37	79
11	—12	4	12	5	49	27	10	6	15	9	26	51
12	—13	3	11	50	85	25	34	4	16	20	35	69
13	—19	4	5	50	850	216	850	3	219	5	13	18
14	—31	4	4	30	230	77	301	6	106	16	54	65
15	—31	4	4	20	50	19	17	4	11	6	22	23
16	—40	4	4	50	24	9	155	5	66	11	32	42
17	—40	4	4	30	50	20	35	5	18	19	45	67
18	—40	4	4	10	174	60	389	6	132	15	40	51
19	—49	4	5	50	111	42	134	7	55	19	66	83
20	—50	4	5	50	*159+	69+	6	4	7	5	14	20
21	—50	4	5	20	3000+	1130+	8	4	8	4	13	16
22	—50	4	5	10	82	28	7	4	8	6	14	20
23	—55	4	8	50	45	17	5	3	7	4	11	19
24	—69	4	7	50	74	28	35	8	23	12	39	58
25	—72	4	12	50	904	494	918	8	498	32	96	193
26	—75	4	7	50	42	20	2	5	7	2	6	9
27	—76	4	7	50	20	9	3000+	3	1213+	18	51	81
28	—84	4	4	50	442	152	145	3	56	7	15	21
29	—90	4	6	50	48	20	9	4	9	6	14	23
30	—92	4	4	50	60	23	57	3	25	23	37	61
Среднее					>805	>389	>393	5,2	>206	11,5	32,5	56,3
Для 70 остальных решенных задач:												
минимальное					0	1	0	0	1	0	0	1
максимальное					30	14	40	7	32	17	27	88
среднее					6,3	3,2	3,7	2,6	5,1	2,9	5,8	10,2
Для всех 100 решенных задач:												
среднее					>246	>119	>120	3,4	>65	5,5	13,8	24,0

* Переполнение. (При вычислении средних использовали достигнутые значения.)

средние данные. В конце таблицы приведены средние данные для всех 100 задач. Для задачи № 2 число итераций и время решения заданы в виде 4000+ и 1952+. Это значит, что работа базисного алгоритма прекращена после 4000 итераций и 1952 секунд работы ЭВМ, причем оптимальное решение еще не получено. Аналогично нужно понимать данные для задач № 3, 21 и 27. При этом по характеру изменения и достигнутому значению целевой функции можно было судить, что для получения оптимальных решений этих задач требуется, вероятно, еще несколько тысяч итераций. В случае задач № 5, 6, 7 и 20, где данные представлены в аналогичном виде, причиной прекращения работы было переполнение из-за появления слишком больших элементов матрицы (использованная ЭВМ оперирует целыми числами в пределах от

—32767 до +32767). Графа «Число вспомогательных итераций» показывает, сколько симплексных итераций потребовалось для решения всех вспомогательных задач (11)—(15). При этом одна вспомогательная итерация требовала в 3—5 раз больше времени, чем основная итерация. Из таблицы видно, что в случаях, когда использование базисного алгоритма требует много времени, ускоренный вариант значительно сокращает его. В то же время, ускоренный вариант иногда требует больше времени, чем базисный алгоритм, хотя в первом случае число основных итераций обычно уменьшается. Что касается задач, которые решались базисным алгоритмом не более чем через 40 итераций, то в случае ускоренного варианта алгоритма наблюдалось почти всегда увеличение времени. Однако в среднем ускоренный вариант алгоритма сократил число основных итераций в 45 раз и время решения в 5 раз.

Главный вывод, который можно сделать на основе проведенного эксперимента, заключается в том, что в случае решенных задач базисный алгоритм вел себя нерегулярно, а ускоренный вариант алгоритма — регулярно. Кроме того, в случае использования ускоренного варианта алгоритма никогда не наблюдалось переполнения. Это, наверное, объясняется небольшим числом основных итераций, требуемых для решения задач (известно, что с увеличением числа итераций элементы матрицы задачи имеют тенденцию возрастать по абсолютной величине). Одним преимуществом ускоренного варианта явилось то, что он всегда обнаруживал неразрешимость задачи, а базисный алгоритм окончательного ответа иногда не давал (вычисления пришлось прекратить после большого числа итераций или же переполнения).

Что касается варианта с ускорением только на первой итерации, то этот вариант часто существенно ускорял решение задач (№ 1, 2, 4, 9, 11, 12 и др.). Однако в некоторых случаях он также сильно увеличивал общее число итераций или даже «портил» задачи (№ 5, 16, 18 и 27). Поэтому этот вариант алгоритма нельзя считать регулярным.

Остановимся еще коротко на других вариантах алгоритма. Использование ускорения на двух, трех или даже n первых итерациях не превращало алгоритма регулярным, хотя уменьшало среднее время и число итераций. Применение ускорения в случаях, когда значение целевой функции не менялось в течение двух или трех последовательных итераций, давало в среднем лучшие результаты, чем базисный алгоритм, и худшие результаты, чем ускоренный вариант алгоритма. Наибольшее время требовалось для решения задачи № 2: соответст-

Задача № 21

$R=-50$, $n=4$, $m=5$, $q=20$

	1	$-x_6$	$-x_7$	$-x_8$	$-x_9$
x_0	6	24	30	35	19
x_1	—7	12	9	2	—12
x_2	—15	—4	0	0	—11
x_3	—5	3	—3	—8	—1
x_4	—10	—20	2	—11	0
x_5	12	9	—7	—5	18

оптимальное решение:

$x^*=(0, 7, 32, 32, 3, 0, 1, 4, 2)$,
$x_0=-202$.

Задача № 28

$R=-84$, $n=4$, $m=4$, $q=50$

	1	$-x_5$	$-x_6$	$-x_7$	$-x_8$
x_0	20	42	17	88	7
x_1	—22	28	—33	—47	—43
x_2	—41	—8	32	—48	41
x_3	—20	—38	—48	—9	—5
x_4	—16	—48	—38	5	39

оптимальное решение:

$x^*=(105, 23, 46, 12, 0, 1, 2, 0)$,
$x_0=-173$.

венно 260 и 395 секунд. К сожалению, при решении задач № 6, 7, 8 и 20 эти варианты показали переполнение уже после нескольких десятков секунд работы ЭВМ. Регулярным оказался вариант алгоритма, при котором ускорением пользовались на двух первых итерациях и всегда тогда, когда значение целевой функции не изменялось в течение двух последовательных итераций. Однако среднее время решения задач № 1—30 в 1,2 раза больше соответствующего показателя для ускоренного варианта алгоритма.

В конце предыдущей страницы в качестве примера приведены матрицы и оптимальные решения двух задач (№ 21 и 28).

ЛИТЕРАТУРА

1. *Гомори Р. Е.* В кн.: Календарное планирование. М., «Прогресс», 1966, 227—240.
2. *Корбут А. А., Финкельштейн Ю. Ю.* Дискретное программирование. М., «Наука», 1969.
3. *Кивистик Л.* Изв. АН ЭССР. Физ. Матем., **33**, № 2, 197—200 (1984).
4. *Финкельштейн Ю. Ю.* Докл. АН СССР, **193**, № 3, 543—546 (1970).
5. *Юдин Д. Б., Гольштейн Е. Г.* Линейное программирование. Теория и конечные методы. М., Изд. физ.-мат. лит., 1963.

Тартуский государственный университет

Поступила в редакцию
13/I 1984

L. KIVISTIK

GOMORY TÄIELIKULT TÄISARVULISE ALGORITMI KIIRENDAMISEST

Artiklis on Gomory täielikult täisarvulise algoritmi kiirendamiseks soovitatud kasutada surrogaatkitsendusi (5), mille kordajad y_i saadakse tingimusest, et simplekstabeli vabaliikmete veerg A_0 kahaneks igal iteratsioonisammul võimalikult palju. Sellised kordajad saab leida lineaarplaneerimise ülesannete (11)—(15) lahendamise teel. Nagu näitas arvutuseksperiment, töötab algoritmi soovitatud variant keskmiselt märksa kiiremini kui Gomory baasalgoritm ja on vastupidi viimasele regulaarne.

L. KIVISTIK

ON ACCELERATING GOMORY'S ALL-INTEGER ALGORITHM

In this paper the Gomory's all-integer algorithm for solving the following programming problem is considered:

$$\max x_0 = a_{00} + \sum_{j \in J} a_{0j}(-x_j)$$

subject to

$$x_i = a_{i0} + \sum_{j \in J} a_{ij}(-x_j) \geqslant 0 \quad (i \in I),$$

$$x_j = 0 + (-1)(-x_j) \geqslant 0 \quad (j \in J),$$

x_j is integer $(j \in I \cup J \cup \{0\})$.

For generating Gomory's cuts the surrogate constraints

$$\sum_{i \in I} a_{i0} y_i + \sum_{j \in J} \left(\sum_{i \in I} a_{ij} y_i \right)(-x_j) \geqslant 0$$

are used. The coefficients $y_i \geqslant 0$ are found from the condition that the column of constant terms of the simplex tableau decreases on each iteration step lexicographically as much as possible. For finding such coefficients it is necessary to solve certain linear programming problems. Finiteness of the received algorithm follows from the author's previous paper [3]. The computational experiment executed by the author showed that this variant of the algorithm works, on the average, more rapidly than Gomory's basic algorithm does and, contrary to the latter, it is regular.