МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1981 год Выпуск 89

УДК 519.854.3

# О ПАРАЛЛЕЛИЗМЕ В ЦЕЛОЧИСЛЕННОЙ ОПТИМИЗАЦИИ

В.И. Хохлюк

В работе рассмотрены ситуации, возникающие при решении задач целочисленной оптимизации и допускающие распараллеливание вычислений. В этих ситуациях используются действия над векторами и матрицами, компоненты которых естественным образом распределяются по ветвям параллельного алгоритма. Это естественное распараллеливание можно усилить различными приемами и построениями [1,2]. В работе предполагается, что размеры задач, по крайней мере, на порядок превосходят число процессоров вычислительной системы. Разделы 1 и 2 работы содержат краткий обзор случаев, когда при решении задач оптимизации [6] возможно распараллеливание. Разделы 3-8 [3-7] составляют оригинальную часть работы. В этих разделах изложены алгоритм для представления общего целочисленного решения одного уравнения, алгоритм нахождения всех крайних лучей конуса, алгоритм нахождения всех вершин многогранника, прямой алгоритм для решения целочисленной линейной задачи, параллельный алгоритм ветвей и границ, параллельный алгоритм для рекуррентных соотношений. Эти алгоритмы могут служить вспомогательными средствами для распараллеливания других алгоритмов.

## 1. Простейшие преобразования

Приведем некоторые обозначения, используемые в работе: $i=\overline{k,l}$ – $i = k,k+1,\ldots, l$ для $k<l$; $i=k,k-1,\ldots,l$ для $k > l$; $i = k$ для $k = l$, где $k,l$ – целые; $\forall i$ – для каждого $i$; $[x]$ – наибольшее целое, не большее $x$; функция – целая часть от $x$ определена для всех действительных $x$; $B^n$ – множество всех $n$-мерных векторов с компонентами $0,1$; $Z^n$ – множество всех $n$-мерных векторов целых чисел; $p$ – число параллельно работающих процессоров.

Для определенности будем рассматривать две задачи оптимизации. Первую из них называют целочисленной задачей, а вторую - смешанной целочисленной линейной задачей.

I. Максимизировать $f(x)$ при условиях $g_i(x)\ \rho_i 0$, $i=\overline{1,m}$, $x\geq 0$, $x \in Z^n$. Здесь $f(x)$, $g_i(x)$ - действительнозначные функции от вектора $x$ переменных $x_1, x_2, \ldots, x_n$, а $\rho_i$ - один из символов: $\leq$, $=$, $\geq$ $(i = \overline{1,m})$.

2. Максимизировать $z(x,v) = c_1x+c_2v$ при условиях $A_1x+A_2v \leq b$, $x \geq 0$, $v \geq 0$, $x \equiv 0 (\mathrm{mod}\, 1)$. Здесь $A_1, A_2$, $b, c_1, c_2$ - заданные матрицы с действительными элементами, имеющие соответственно размеры $m \times n_1$, $m \times n_2$, $m \times 1$, $1 \times n_1$, $1 \times n_2$.

Для того чтобы исходные задачи оптимизации свести к некоторым стандартным задачам, для которых, в частности, могут быть готовы программные средства, используют различные преобразования. Здесь под преобразованием задачи оптимизации будем понимать какую-либо операцию над ее элементами (переменными, ограничениями, целевой функцией, множеством допустимых решений), в результате выполнения которой получается новая задача оптимизации. После выполнения преобразования задачи могут измениться число переменных, число и вид ограничений, целевая функция, множество допустимых решений, условие целочисленности; наконец, преобразованная задача может быть сформулирована в терминах, отличных от терминов исходной задачи.

Рассмотрим несколько простейших преобразований задачи, которые естественным образом распараллеливаются. Параметром, по которому производится распределение данных задачи между $p$ параллельно работающими процессорами, может служить либо число ограничений, либо число переменных, либо число строк или столбцов матрицы, либо длина однородного массива и т.д. При этом, как правило, получаются параллельные ветви, слабо связанные между собой или совсем не связанные. Таким образом, это дает ускорение счета в $p$ раз. Целесообразность того или иного распределения данных определяется вычислениями, которым предшествуют операции преобразований.

Переход от начальной задачи к конечной может состоять из нескольких последовательно выполняемых преобразований.

I. Переход к неотрицательным переменным. Переменную $x_j$, для которой $l_j \leq x_j \leq u_j$ ($l_j$, $u_j$ - произвольные числа), можно заменить либо переменной $x'_j$, изменяющейся в интервале $[0, u'_j]$, либо переменной $x''_j$ со значения-

ми из интервала [0,1]. Формулами преобразования будут: $x_j = x'_j + l_j$, $0 \le x'_j \le u'_j = u_j - l_j$; $x_j = (u_j - l_j)x''_j + l_j$, $0 \le x''_j \le 1$. Для $u_j = +\infty$ и $l_j = -\infty$ будем иметь соответственно $x_j = x'_j + l_j$, $0 \le x'_j < +\infty$, и $x_j = -x'_j + u_j$, $0 \le x'_j < +\infty$. Для $l_j = -\infty$, $u_j = +\infty$ переменная $x_j$ произвольного знака заменяется двумя неотрицательными переменными, а именно: $x_j = x_j^+ - x_j^-$, $x_j^+ \cdot x_j^- = 0$, $0 \le x_j^+ < +\infty$, $0 \le x_j^- < +\infty$, где $x_j^+ = \max(0, x_j)$, $x_j^- = \max(0, -x_j)$. Для каждого базисного решения линейной задачи оптимизации нелинейное условие вида $x_j^+ \cdot x_j^- = 0$ выполняется, так как переменные $x_j^+$ и $x_j^-$ не могут быть одновременно базисными.

2. Переход к бинарным переменным. Целочисленная переменная $x_j$, для которой $0 \le x_j \le u_j$, заменяется системой бинарных переменных. Эту замену можно осуществить двумя различными способами:

$$x_j = \sum_{k=0}^{u_j} k y_{j,k}, \quad \sum_{k=0}^{u_j} y_{j,k} = 1, \quad y_{j,k} = 0,1 \ (k = \overline{0,u_j});$$

$$x_j = \sum_{k=0}^{t_j} 2^k y_{j,k} \le u_j, \quad y_{j,k} = 0,1 \ (k = \overline{0,t_j}),$$

где $2^{t_j} \le u_j < 2^{t_j+1}$. Первый способ требует $u_j+1$ бинарных переменных, а второй $-[\log_2 u_j] + 1$.

3. Преобразование ограничений. Уравнение $g_i(x)=0$ эквивалентно двум неравенствам $g_i(x) \le 0$, $g_i(x) \ge 0$; вводя слабые переменные $s_i \ge 0$ и $t_i \ge 0$, можно преобразовать неравенства $g_i(x) \le 0$ и $g_i(x) \ge 0$ соответственно в уравнения $g_i(x) + s_i = 0$ и $g_i(x) - t_i = 0$.

4. Преобразование линейных ограничений. Систему из $m$ линейных уравнений можно заменить системой из $m+1$ линейных неравенств, так как

$$\{x \mid Ax = b\} = \{x \mid Ax \ge b, \sum_{i=1}^{m} \left( \sum_{j=1}^{n} a_{i,j} x_j - b_i \right) \le 0\}.$$

5. Агрегация ограничений. Рассмотрим систему ограничений $a_1 x - b_1 = 0$, $a_2 x - b_2 = 0$, $x \in B^n$, где строки $a_1$, $a_2$ и свободные члены $b_1, b_2$ целочисленны. Положим

$$a^-_{1,j} = \min(0, a_{1,j}) \ \forall j, \quad a^+_{1,j} = \max(0, a_{1,j}) \ \forall j,$$

$$\lambda^- = \sum_{j=1}^{n} a^-_{1,j} - b_1, \quad \lambda^+ = \sum_{j=1}^{n} a^+_{1,j} - b_1.$$

Определим $\lambda = \max(|\lambda^-|, |\lambda^+|)$ и рассмотрим агрегированное уравнение $(a_1 + \alpha a_2)\, x - b_1 - \alpha b_2 = 0$, где $\alpha$ – произвольное целое число, удовлетворяющее неравенству $|\alpha| > \lambda$.

Очевидно, что любой бинарный вектор $x'$, удовлетворяющий двум уравнениям, удовлетворяет и агрегированному уравнению. Если $x'$ является решением агрегированного уравнения, то $a_1 x' - b_1 = -\alpha(a_2 x' - b_2)$. Левая часть представляет собой целое число, кратное $\alpha$ и по модулю меньшее $|\alpha|$ (в силу выбора $\alpha$ и целочисленности данных). Следовательно, $a_1 x' - b_1 = 0$. Это означает, что $x'$ удовлетворяет системе из двух уравнений. Более общий случай агрегации ограничений описан, например, в [6].

6. Переход к неотрицательной матрице ограничений. Рассмотрим систему ограничений $Ax = b$, $0 \le x \le u$. Если в столбце $a_j$ матрицы A существует элемент $a_{i,j} < 0$ для некоторого $i$, то заменяем $a_j x_j$ на $a'_j x_{j,1} + a''_j x_{j,2}$, где $a'_{i,j} = \min(0, a_{i,j})\,\forall i$, $a''_{i,j} = \max(0, a_{i,j})\,\forall i$. Затем добавляем ограничение $x_{j,2} - x_{j,1} = 0$. Используя преобразование $x_{j,1} = u_j - y_{j,1}$ для всех столбцов $a_j$ рассматриваемого вида, получаем неотрицательную матрицу ограничений.

7. Переход к бинарной матрице ограничений. В системе ограничений $Ax = b$, $x_j = 0,1\ \forall j$, где A – неотрицательная матрица, заменяем каждый столбец $a_j$ матрицы A на $m \times (\sum_{i=1}^{m} a_{i,j})$-матрицу $A'_j = (a'_{i,k})$, где

$$a'_{i,k} = \begin{cases} 0, & 1 \le k \le t_{i-1,j} \vee t_{i,j} < k \le t_{m,j}; \\ 1, & t_{i-1,j} < k \le t_{i,j} \end{cases} \quad (i=\overline{1,m};\ k=\overline{1,t_{m,j}}).$$

Здесь $t_{0,j} = 0$, $t_{i,j} = \sum_{r=1}^{i} a_{r,j}$ $(i=\overline{1,m})$. Таким образом, если $a_j = \begin{pmatrix} 2 \\ 0 \\ 3 \end{pmatrix}$, то

$$A'_j = \begin{pmatrix} I & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I & I & I \end{pmatrix}.$$

Заметим, что единичный вектор не требует преобразования.

Свяжем переменную $x_{j,k}$ с k-м столбцом матрицы $A'_j$. Новая система ограничений эквивалентна первоначальной, если $x_j = x_{j,k}$ $\forall k$. Это условие достигается добавлением ограничений $x_{j,k} + x'_{j,k} = 1$, $x_{j,k+1} + x'_{j,k} = 1$, $x'_{j,k} = 0,1$ $(k = \overline{1, t_{m,j} - 1})$. Полученная бинарная матрица в каждом столбце содержит не больше трех единиц.

Нетрудно показать изменение целевой функции в преобразованиях I-7.

## 2. Преобразования задач оптимизации

I. Задачи о покрытии, разбиении и упаковке множества. Прежде всего отметим, что можно априори уменьшить размеры рассматриваемых задач исключением из матрицы A, задающей ограничения, ее некоторых строк и столбцов. Правила исключения просты и распараллеливаются [6].

Рассмотрим бинарную линейную задачу $\max\{cx \mid Ax=b,\ x_j=0,1\ \forall j\}$, где $b \geq 0$. Эту задачу можно преобразовать в задачу о разбиении $\max\{c'y \mid A'y=1,\ y_j = 0,1\ \forall j\}$, в которой каждый столбец бинарной матрицы ограничений A' содержит не больше трех единиц. Это преобразование состоит из трех преобразований, подобных 6,7 из раздела I [6].

Задача о разбиении преобразуется в задачи о покрытии и упаковке путем замены целевой функции c'y на c"y. Чтобы вычислить элемент $c''_j$ строки c", нужны столбец $a'_j$ матрицы ограничений A' и константа, общая для всех столбцов.

Бинарную линейную задачу

$$\min\{cx \mid \sum_{j=1}^{n} a_{i,j} x_j \geq b_i,\ i = \overline{1,m},\ x_j = 0,1,\ j = \overline{1,n}\}$$

с неотрицательной матрицей $A=(a_{i,j})$ можно преобразовать в задачу о покрытии $\min\{cx \mid A'x \geq 1,\ x_j = 0,1,\ j = \overline{1,n}\}$. В этом преобразовании каждое неравенство системы ограничений заменяется эквивалентной ему системой неравенств с коэффициентами 0,I.

2. Преобразование бинарной полиномиальной задачи в бинарную линейную задачу. Рассмотрим бинарную полиномиальную задачу

$$\max\{p_0(x) \mid p_i(x) \rho_i 0,\ i = \overline{1,m},\ x_j = 0,1,\ j = \overline{1,n}\},$$

где $p_i(x)(i = \overline{0,m})$ - заданные полиномы от n бинарных переменных. Предположим, что степень хотя бы одного из многочленов $p_i(x)$ $(i=\overline{0,m})$ больше I. Опустив коэффициент, рассмотрим произвольный член полинома $\prod_{j\in J} x_j$, где $J = \{j_1, j_2, \ldots, j_{|J|}\}$ - множество индексов бинарных переменных, входящих в этот член ($|J| \geq 2$). Рассматриваемый член равен I, если $x_j=1$ для всех $j \in J$, и равен 0, если это не так.

Введем бинарную переменную $z_J$ и два неравенства

$$\sum_{j\in J} x_j - z_J \leq |J| - 1, \quad -\sum_{j\in J} x_j + |J|\, z_J \leq 0 .$$

Легко проверить соотношения

$$\prod_{j\in J} x_j = 1 \longleftrightarrow z_J = 1, \quad \prod_{j\in J} x_j = 0 \longleftrightarrow z_J = 0 .$$

Заменяя каждое нелинейное слагаемое $a_J \prod_{j\in J} x_j$ на $a_J z_J$, получаем бинарную линейную задачу, в которой появляются новые переменные $z_J$ и новые ограничения указанного вида.

Существуют различные приемы для уменьшения числа переменных, появляющихся в процессе линеаризации. Они особенно важны для практических задач.

3. Линейная задача оптимизации. При решении линейной задачи $\max\{cx \mid Ax=b,\ x \geq 0\}$ симплекс-методом возможны два подхода к распараллеливанию вычислений. Во-первых, можно распараллелить выбор небазисного столбца для введения в базис, во-вторых, - преобразование обратной базисной матрицы.

4. Алгоритмы ветвей и границ целочисленной оптимизации. Многие из этих алгоритмов содержат этапы, на каждом из которых нужно осуществить выбор из p альтернатив, почти одинаковых по количеству выполняемых операций. Параллельное выполнение таких альтернатив может обеспечивать ускорение счета в p раз. В разделе 7 настоящей работы описан p-ветвевой обход дерева перебора [7].

5. Алгоритмы отсечения целочисленной оптимизации. Распараллеливание этих алгоритмов основывается на возможности распараллелить умножение двух матриц [I]. Учет конкретной специфики алгоритмов отсечения может только усилить эффект от процесса распараллеливания [7].

6. Преобразование целочисленной матрицы. В алгоритмах приведения целочисленной матрицы к

диагональному и трапецеидальному видам целесообразно совокупность всех целых чисел представить в виде однородного бинарного массива. Обработку такого бинарного массива, который может быть весьма большим, рационально распределять между параллельно работающими процессорами. Второй способ распараллеливания состоит в разбиении данной матрицы A на клетки, в приведении ее отдельных клеток к нужному виду и выполнении преобразований, дающих диагональную или трапецеидальную матрицу.

## 3. Представление общего целочисленного решения одного уравнения

В разделах 3–5 описано решение трех задач, алгоритм каждой из которых может быть использован для отдельных ветвей при параллельном решении более сложных задач оптимизации.

Здесь будет показано, как можно представить общее целочисленное решение одного уравнения при помощи $2n$ чисел вместо $n^2$, выполнив небольшое число арифметических операций.

Рассмотрим уравнение $\sum_{j=1}^{n} a_j \xi_j = a_0$, в котором заданные коэффициенты $a_j$ $(j=\overline{1,n})$ отличны от нуля и целочисленны, а свободный член $a_0$ – целое число.

Пусть $d_n$, $x_j (j=\overline{1,n})$, $y_2, z_2, u_2, v_2, k, l, \bar{y}_3$ – величины, удовлетворяющие условиям

$$y_2 a_k + z_2 a_l = d_2; \quad u_2 a_k + v_2 a_l = 0; \quad \bar{y}_3 d_2 + \sum_{\substack{j=1 \\ j \neq k,l}}^{n} x_j a_j = d_n.$$

Здесь $d_n$ – наибольший общий делитель чисел $a_j (j=\overline{1,n})$, а $x_j (j=\overline{1,n})$ – его целочисленные коэффициенты разложения.

Предполагаем, что $k=1$, $l=2$, $x_j \neq 0$ $(j=1,s)$, $x_j = 0$ $(j=\overline{s+1,n};\ s \geq 1)$. Это предположение не нарушает общности, так как оно может быть достигнуто перенумерацией переменных. Считаем, что величина $d_n$ является делителем числа $a_0$, иначе уравнение не имеет целочисленных решений.

Рассмотрим матрицу $A = \begin{pmatrix} a \\ I_n \end{pmatrix}$. Здесь $a = (a_1, a_2, \ldots, a_n)$ – строка заданных целых чисел, $I_n$ – единичная матрица $n$-го порядка. Ниже, используя только столбцовые преобразования, в три этапа

строится унимодулярная целочисленная матрица K , задающая преобразование переменных $\xi = K\zeta$. Матрица K получается на месте единичной матрицы $I_n$ . При выполнении преобразований ограничимся рассмотрением только строки a .

Над столбцами 1 и 2 выполняем преобразование вида $(a_1\ a_2)\begin{pmatrix} u_2 & y_2 \\ v_2 & z_2 \end{pmatrix} = (0\ d_2)$. Как хорошо известно, целочисленная матрица $\begin{pmatrix} u_2 & y_2 \\ v_2 & z_2 \end{pmatrix}$ унимодулярна.

Теперь над столбцами выполняем преобразования, которые для рассматриваемой строки дают $1\cdot 0 + \bar{y}_3 d_2 + \sum_{j=3}^{s} x_j a_j = d_n$. Это означает, что к первому столбцу прибавляются $s-1$ столбцов, умноженных на соответствующие коэффициенты. Наконец, вычитаем из 2-го, 3-го,..., n-го столбца последней матрицы ее первый столбец, умноженный соответственно на $(d_2/d_n), (a_3/d_n), \ldots, (a_n/d_n)$. В результате получаем строку $(d_n, 0, \ldots, 0)$, а под ней матрицу K . Очевидно, что матрица K целочисленна и унимодулярна.

Обозначим столбцы матрицы K через $k_1, k_2, \ldots, k_n$. Тогда общее целочисленное решение рассматриваемого уравнения имеет вид $\xi = (a_0/d_n)k_1 + \zeta_2 k_2 + \zeta_3 k_3 + \ldots + \zeta_n k_n$; $\zeta_2, \zeta_3, \ldots, \zeta_n$ – произвольно изменяющиеся целочисленные переменные. Это решение легко записать покомпонентно. Например, $\xi_1 = (a_0/d_n)(u_2 + \bar{y}_3 y_2) + \zeta_2(y_2 - (d_2/d_n) \times (u_2 + \bar{y}_3 y_2)) - \zeta_3(a_3/d_n)(u_2 + \bar{y}_3 y_2) - \ldots - \zeta_n(a_n/d_n)(u_2 + \bar{y}_3 y_2)$.

Таким образом, для представления общего целочисленного решения одного уравнения нужны величины $a_j$ $(j = \overline{0,n})$, $d_n$, $x_j$ $(j = \overline{1,s};\ s \le n)$, а также дополнительная информация из семи чисел $y_2$, $z_2$, $u_2$, $v_2$, k, l, $\bar{y}_3$, которой можно пренебречь. Итак, требуется не больше, чем $2n+1$ чисел. Более того, как правило, величина s мала по сравнению с n . Это подтверждается счетом на ЭВМ и согласуется с теорией.

## 4. Нахождение всех крайних лучей конуса

Рассмотрим заостренный конус $C = \{ z \mid z_j \ge 0,\ j = \overline{1,n},\ \sum_{j=1}^{n} a_{i,j} z_j \ge 0,\ i = \overline{1,m} \}$. Здесь все элементы $a_{i,j}$ матрицы A – действительные числа $(i = \overline{1,m};\ j = \overline{1,n})$. Требуется найти направляющие векторы единичной длины всех крайних лучей конуса C . Та-

кую систему векторов будем называть системой крайних векторов. Под длиной вектора $a = (a_1, a_2, \ldots, a_n)$ понимаем величину $|a| = \sqrt{\sum_{j=1}^{n} a_j^2}$.

Укрупненный шаг k алгоритма состоит в том, что из системы крайних векторов конуса, заданного n+k-1 неравенствами, получается система крайних векторов $x_1, x_2, \ldots, x_r$ нового конуса, задаваемого теми же неравенствами и еще одним добавленным неравенством ($k = \overline{1,m}$). Каждому крайнему вектору $x_i$ соответствует бинарная строка $l_i = (l_{i,1}, l_{i,2}, \ldots, l_{i,h})$, где

$$l_{i,t} = \begin{cases} 1, & a_t x_i > 0, \\ 0, & a_t x_i = 0 \end{cases} \quad (i=\overline{1,r};\ t=\overline{1,h};\ h \le k).$$

Здесь $a_t$ – t -я строка матрицы A, h – число добавленных неравенств, менявших систему крайних векторов.

Работа алгоритма начинается с конуса, заданного условиями неотрицательности $x_j \ge 0$ ($j=\overline{1,n}$).

А л г о р и т м.

Шаг 1. r:= n, h:= 0, k:= 1, $x_i := e_i$ ($i=\overline{1,r}$; $e_i$-i-й единичный вектор).

Шаг 2. Если $k > m$, то переходим на шаг 7. Выбираем строку $a_k$ матрицы A и нормируем ее $a_k := a_k/|a_k|$.

Шаг 3. Вычисляем скалярные произведения $b_i = a_k x_i$ ($i = \overline{1,r}$). Если $b_i \ge 0$ ($i = \overline{1,r}$), то k:= k+1, и переходим на шаг 2. Если же $b_i < 0$ ($i = \overline{1,r}$), то r:=0, и переходим на шаг 7.

Шаг 4. Для каждой пары (i,j) такой, что $b_i > 0$, $b_j < 0$, подсчитываем число позиций, в которых стоят нули в обеих строках

$$(x_{i,1}, x_{i,2}, \ldots, x_{i,n}, l_{i,1}, l_{i,2}, \ldots, l_{i,h})$$

$$(x_{j,1}, x_{j,2}, \ldots, x_{j,n}, l_{j,1}, l_{j,2}, \ldots, l_{j,h}).$$

Если это число больше или равно n – 2, то проверяем, существует ли среди строк $x_{s,1}, x_{s,2}, \ldots, x_{s,n}, l_{s,1}, l_{s,2}, \ldots, l_{s,h}$ ($s = \overline{1,r}$; $s \ne i$, $s \ne j$) строка, имеющая нули в тех же позициях. При отсутствии такой строки в систему новых крайних векторов заносим вектор $x' = \alpha x_i + \beta x_j$. Коэффициенты $\alpha, \beta$ выбираем из условий $\alpha > 0, \beta > 0$, $\alpha b_i + \beta b_j = 0$, $|x'| = 1$. Соответствующая вектору x' бинарная строка $l' = (l'_1, l'_2, \ldots, l'_h)$ образуется по правилу

$$l'_t = \begin{cases} 1, & l_{i,t} = 1 \vee l_{j,t} = 1; \\ 0, & l_{i,t} = 0 \wedge l_{j,t} = 0 \end{cases} \quad (t=\overline{1,h}).$$

Шаг 5. Присоединяем к системе новых крайних векторов те из старых, для которых $b_i \geq 0$, и получаем систему крайних векторов $x'_1, x'_2, \ldots, x'_{r'}$ конуса, заданного $n+h+1$ неравенствами.

Шаг 6. Добавляем в каждую бинарную строку $i$ новой системы элемент $l_{i,h+1}$. Для вновь образованных векторов $l_{i,h+1} = 0$, для старых –

$$l_{i,h+1} = \begin{cases} 1, & b_i > 0, \\ 0, & b_i = 0. \end{cases}$$

Заменяем $h$ на $h+1$, $k := k+1$, опускаем штрихи при $x'_i$ и $r'$ и переходим на шаг 2.

Шаг 7. Работа алгоритма закончена. Система крайних векторов $x_1, x_2, \ldots, x_r$ конуса $C$ найдена. При $r = 0$ существует только нулевое решение рассматриваемой системы неравенств.

ЗАМЕЧАНИЕ 1. Коэффициенты $\alpha, \beta$ вычисляются по формулам

$$\alpha = \sqrt{\frac{b_j^2}{b_j^2 + 2b_j b_i x_j^T x_i + b_i^2}}, \qquad \beta = \frac{-\alpha\, b_i}{b_j}.$$

ЗАМЕЧАНИЕ 2. Столбец $t$ и далее сохраняет свое строение, если на шаге $k$ для некоторого фиксированного $t$ $(1 \leq t \leq h)$ среди элементов $l_{i,t}$ $(i = \overline{1,r})$ находятся 0,1 или $r$ нулей. Столбцы таких типов можно исключить из дальнейшего рассмотрения.

ЗАМЕЧАНИЕ 3. Нетрудно геометрически истолковать все шаги алгоритма. Алгебраическое обоснование алгоритма, распределение памяти, организация счета и другие вычислительные детали, весьма важные для машинной реализации, приведены в работах [4,5].

## 5. Нахождение всех вершин многогранника

Рассмотрим линейную задачу $\min\{cx \mid Ax = b,\ x \geq 0\}$, где элементы $a_{i,j}$ $m \times n$-матрицы $A$, компоненты $m$-столбца $b$ и $n$-строки $c$ есть заданные действительные числа. Нужно найти все вершины многогранника $P = \{x \mid Ax = b,\ x \geq 0\}$, расположить их по возрастанию значений целевой функции $cx$. Каждая вершина $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})$ характеризуется записью, состоящей из значения $e_i = cx_i$ целевой функции и бинарной строки $l_i = (l_{i,0}, l_{i,1}, l_{i,2}, \ldots, l_{i,m})$. Здесь

$$l_{i,j} = \begin{cases} 1, & x_{i,j} > 0, \\ 0, & x_{i,j} = 0 \quad (j=\overline{1,n}). \end{cases}$$

Величина $l_{i,0} = 1$ означает, что все вершины, смежные с вершиной $x_i$, уже получены и их записи занесены в список, иначе $l_{i,0} = 0$.

А л г о р и т м.

Шаг I. Двухфазным симплекс-методом решаем линейную задачу $\min \{cx \mid Ax = b;\ x \geq 0\}$. После первой фазы из условий задачи исключаем линейно-зависимые уравнения. В список под номером I заносим запись вершины, являющейся уже найденным оптимальным базисным решением линейной задачи. Объявляем вершину $x_1$ текущей, обозначая ее $x'$, и переходим на шаг 4.

Шаг 2. Среди записей списка, удовлетворяющих условиям $c_i \geq c'$ и $l_{i,0} = 0$, выбираем запись $i_0$ с наименьшим значением $c_{i_0}$, если таких записей нет, то переходим на шаг 6.

Шаг 3. Исходя из текущей вершины $x'$, для которой известна обратная базисная матрица, при помощи конечного числа симплекс-преобразований (преобразований с центральным элементом) получаем обратную базисную матрицу для вершины $x_{i_0}$. Объявляем вершину $x_{i_0}$ текущей.

Шаг 4. Для невырожденной текущей вершины $x'$ последовательно вводим в базис все не базисные столбцы, находим все смежные с $x'$ вершины $x$ со значениями $cx \geq cx'$ и заносим их записи под соответствующими номерами в список, если их там еще не было. Полагаем $l'_0 = 1$ и переходим на шаг 2.

Шаг 5. Для вырожденной текущей вершины $x'$ сначала получаем все ребра многогранника $P$, выходящие из $x'$, затем по этим ребрам находим все смежные с $x'$ вершины $x$ со значениями $cx \geq cx'$ и заносим их записи под соответствующими номерами в список, если их там еще не было. Полагаем $l'_0 = 1$ и переходим на шаг 2.

Шаг 6. Все вершины многогранника $P$ получены. Алгоритм заканчивает свою работу.

Рассмотрим более детально отдельные шаги алгоритма.

На шаге 3 переход от базиса $B'$ текущей вершины $x'$ к базису $B$ вершины $x$ осуществляется выделением столбцов базиса $B$, которые соответствуют положительным базисным переменным и находятся вне базиса $B'$, последовательным вводом этих столбцов в базис и выводом из него столбцов, соответствующих нулевым компонентам вершины $x$. Возможность такого перехода опирается на линейную независимость базисных векторов.

Шаги 4 и 5. Пусть задан базис B вершины x многогранника $P = \{x \mid Ax = b,\ x \geq 0\}$. Покажем, как построить все ребра многогранника P, выходящие из вершины x. Разобьем матрицу A и вектор x на две части $A = (B,N)$, $x = (x_B, x_N)$. Рассмотрим многогранник $P' = \{x_N \mid B^{-1}b - B^{-1}N x_N \geq 0,\ x_N \geq 0\}$. Вершине $x = (B^{-1}b, 0)$ многогранника P соответствует вершина $x_N = 0$ многогранника P', и наоборот. Поэтому достаточно найти все ребра, выходящие из вершины $x_N = 0$ многогранника P'. Для невырожденной вершины $x = (B^{-1}b, 0)$, т.е. при $B^{-1}b > 0$ ребра многогранника P' будут направлены по единичным ортам, соответствующим переменным $x_N$. Для вырожденной вершины, т.е. при $I = \{i \mid (B^{-1}b)_i = 0\} \neq \emptyset$, ребра, выходящие из вершины $x_N = 0$ многогранника P', будут лежать на крайних лучах конуса $C = \{x_N \mid (-B^{-1}N x_N)_i \geq 0,\ i \in I,\ x_N \geq 0\}$. Для разыскания этих крайних лучей и применяется описанный выше алгоритм.

Данный алгоритм отличается от других алгоритмов для нахождения всех вершин многогранника (например, из [8]) тем, что перебор осуществляется по вершинам, а не по их базисам. Для вырожденной вершины может существовать огромное количество представляющих ее базисов. Приведем пример, иллюстрирующий такую ситуацию.

ПРИМЕР. Рассмотрим линейную задачу $\min\{\sum_{j=1}^{n} c_j x_j \mid x_{j_1} + x_{j_2} \leq 1,\ 1 \leq j_1 < j_2 \leq n,\ x_j \geq 0,\ j = \overline{1,n}\}$. Она содержит $\frac{1}{2}n(n-1)$ основных ограничений. Для нечетного целого n вершина $x = (\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ многогранника условий задачи имеет не менее, чем $\frac{1}{2}(n-1)!$ представляющих ее базисов. Действительно, определитель матрицы коэффициентов системы неравенств $x_{j_r} + x_{j_{r+1}} \leq 1$ $(r = \overline{1,n};\ j_{n+1} = j_1)$ для нечетного n равен 2. Здесь $(j_1, j_2, \ldots, j_n)$ – перестановка чисел 1, 2, ..., n. Заметим, что 2n перестановок

$$(j_1, j_2, \ldots, j_n), (j_2, j_3, \ldots, j_n, j_1), \ldots,$$

$$(j_n, j_1, j_2, \ldots, j_{n-1}), (j_n, j_{n-1}, \ldots, j_1),$$

$$(j_{n-1}, j_{n-2}, \ldots, j_1, j_n), \ldots, (j_1, j_n, j_{n-1}, \ldots, j_2)$$

задают одну и ту же систему неравенств. Таким образом, число представлений рассматриваемого вида для вершины x будет равно $\frac{n!}{2n} = \frac{1}{2}(n-1)!$ Легко указать базис для рассматриваемой вершины, если перейти от неравенств к уравнениям.

Описанный здесь алгоритм для нахождения всех вершин многогранника был реализован на языке АЛГОЛ. Практически находились все вершины из некоторой окрестности оптимальной вершины. Этот подход позволил решить ряд конкретных задач, которые не удавалось решить другими способами (например, из [5]).

## 6. Прямой алгоритм для решения целочисленной линейной задачи

Этот алгоритм будет изложен на геометрическом языке. При этом некоторые детали останутся вне рассмотрения. Возможность распараллеливания алгоритма обусловлена распараллеливаемостью его составных частей.

Рассмотрим целочисленную линейную задачу $\min \{cx \mid A'x \geq b';\ x \geq 0,\ x \equiv 0\}$. Здесь $m \times n$ – матрица $A'$, $m$ – столбец $b'$ и $n$ – строка $c$ состоят из заданных целых чисел. Считаем, что $b' \leq 0$. Через $P$ обозначим выпуклую оболочку допустимых решений этой целочисленной линейной задачи, т.е. выпуклую оболочку множества точек $\{x \mid Ax \geq b,\ x \equiv 0\}$, где $A = \begin{pmatrix} I_n \\ A' \end{pmatrix}$, $b = \begin{pmatrix} 0 \\ b' \end{pmatrix}$. Прямой алгоритм состоит в построении конечной последовательности целочисленных вершин многогранника $P$, которая приводит к минимуму.

Предположим, что на некотором шаге алгоритма известны целочисленная вершина $x'$ многогранника $P$ и правильный конус $C' = \{x \mid D'x \geq f',\ |D'| \neq 0\}$ с вершиной в точке $x'$, $P \subseteq C'$. В систему $D'x \geq f'$ входят некоторые неравенства из условий задачи $Ax \geq b$, а также, быть может, неравенства, построенные на предыдущих шагах алгоритма. Матрица $D'$ и вектор $f'$ целочисленны.

Пусть система целочисленных векторов $u_1, u_2, \ldots, u_n$ задает все крайние направления конуса $C'$.

Достаточное условие оптимальности: целочисленная точка $x'$ является оптимальным решением целочисленной линейной задачи, если $cu_k \geq 0$ для $k = \overline{1,n}$.

Необходимое и достаточное условие оптимальности: для всех крайних направлений $u'_k$, выходящих из вершины $x'$ многогранника $P$, выполняются неравенства $cu'_k \geq 0$ $(k = \overline{1,n'})$.

Теперь поиск направления, сдвиг вдоль которого приводит к лучшему решению (такое направление будем называть улучшающим), можно организовать, например, следующим образом. Среди векторов

$u_1, u_2, \ldots, u_n$ ищем направление $u_k$, удовлетворяющее условиям: а) $cu_k < 0$; б) $a_i u_k \geq 0$, $i \in I$, где $I = \{i \mid a_i x' = b_i\}$; в) на луче $x(\theta) = x' + \theta u_k$, $\theta > 0$, существует по крайней мере одно допустимое решение целочисленной линейной задачи.

Если улучшающее направление не найдено, то нужно перейти к конусу $\tilde{C} = \{x \mid D'x \geq f', a_i x \geq b_i, i \in I\}$ и для его всех крайних направлений последовательно проверить условия "а", "в".

Если же и в этом случае улучшающее направление не найдено, то рассмотрим конус $\tilde{\tilde{C}} = \{x \mid D'x \geq f', a_i x \geq b_i, i \in I, \pi_i x \geq \pi_0, i \in I'\}$ с вершиной в точке $x'$. Неравенства $\pi_i x \geq \pi_0$, $i \in I'$, задают грани соответствующего углового многогранника, которые проходят через точку $x'$ и не были еще построены. Вопрос о том, как получать нужные грани, выходит за рамки данного рассмотрения.

При переходе от $C'$ к $\tilde{C}$ и от $\tilde{C}$ к $\tilde{\tilde{C}}$ изменяется система крайних векторов, задающая соответствующий конус. Если для всех крайних направлений некоторого конуса $\tilde{\tilde{C}}$ не выполняется условие "а", то точка $x'$ является оптимальным решением. Неравенство $cu_k < 0$ приводит либо к новой целочисленной точке, либо к исключению направления $u_k$. Этот процесс конечен.

Остановимся более детально на условии "в". Пусть для некоторого целочисленного вектора $u_k$ из системы $u_1, u_2, \ldots, u_n$ выполняются условия "а" и "б". Рассмотрим точки $x(\theta) = x' + \theta u_k$, $\theta \geq 0$. Если точки $x(\theta)$ для любого $\theta \geq 0$ удовлетворяют системе $Ax \geq b$, то значения целевой функции не ограничены снизу на множестве целочисленных точек $\{x(k) \mid k = \overline{1, +\infty}\}$. Определим величину $\theta_0$ из соотношения

$$\theta_0 = \min_{a_i u_k < 0} \frac{a_i x' - b_i}{-a_i u_k}.$$

Через $l$ обозначим номер неравенства, где достигается минимум $(1 \leq l \leq n + m)$. Таких $l$ может быть несколько.

Пусть $\theta'$ - наибольшее значение $\theta$ из интервала $[0, \theta_0]$, для которого $x(\theta')$ является целочисленной точкой. Если $\theta' = 0$, то такое направление не является крайним направлением многогранника $P$ и не может быть улучшающим направлением.

Пусть $x(\theta') = x''$ - новая вершина многогранника $P$ с меньшим значением целевой функции. Рассмотрим правильный конус $C'' = \{x \mid D''x \geq f'', |D''| \neq 0\}$ с вершиной в точке $x''$. Этот конус задается $n-1$ неравенствами, левые части которых линейно-независимые и обращаются вектором $u_k$ в нуль, а также гранью углового многогранника, соответствующего конусу с вершиной в точке $x(\theta_0)$.

ПРИМЕР. Минимизировать $-x_1-2x_2$ при условиях $x_1+x_2\le 7$, $2x_1\le 11$, $2x_2\le 7$, $x_1\ge 0$, $x_2\ge 0$, $x_1\equiv 0 \pmod 1$, $x_2\equiv 0 \pmod 1$. На рис.I область, обведенная волнистой линией, изображает выпуклую оболочку P допустимых решений этой конкретной целочисленной линейной задачи.

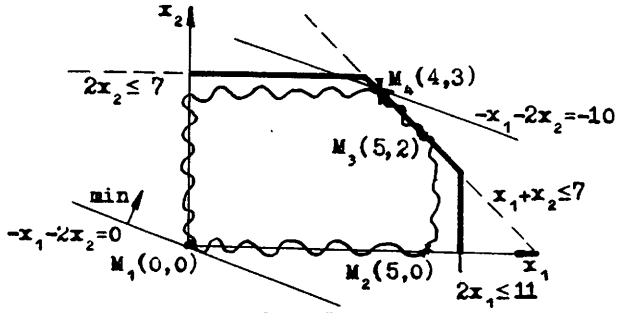


Рис. I

Начинаем с вершины $M_1(0,0)$ многогранника P и правильного конуса $x_1\ge 0$, $x_2\ge 0$. Среди двух улучшающих направлений (I,0) и (0,I), выходящих из точки $M_1$, выбираем направление (I,0). Угловым многогранником для конуса $2x_1\le 11$, $x_2\ge 0$ служит многогранник $x_1\le 5$, $x_2\ge 0$. Таким образом, сдвиг вдоль направления (I,0) приводит в точку $M_2(5,0)$ и правильному конусу $x_1\le 5$, $x_2\ge 0$.

Среди направлений (-I,0) и (0,I), выходящих из точки $M_2$, улучшающим является направление (0,I). Сдвиг вдоль этого направления приводит в точку $M_3(5,2)$ и к правильному конусу $x_1\le 5$, $x_1+x_2\le 7$.

Среди направлений (0,-I) и (-I,I), выходящих из точки $M_3$, улучшающим является направление (-I,I). Угловым многогранником для конуса $x_1+x_2\le 7$, $2x_2\le 7$ служит многогранник $x_1+x_2\le 7$, $x_2\le 3$. Сдвиг вдоль направления (-I,I) приводит в точку $M_4(4,3)$ и к правильному конусу $x_1+x_2\le 7$, $x_2\le 3$.

Направления (I,-I) и (-I,0), выходящие из точки $M_4$, не являются улучшающими, поскольку целевая функция вдоль этих направлений возрастает, следовательно, точка $M_4(4,3)$ является оптимальным решением со значением -I0.

Все рассуждения и построения опираются на рис.I.

## 7. Параллельный алгоритм ветвей и границ

В упрощенной ситуации будет показано, как можно осуществить p-ветвевой обход дерева перебора. При решении, например, бинарной линейной задачи максимизации, когда все время уточняются нижняя и

верхняя границы для значений целевой функции, использование информации о границах сразу в p ветвях может ускорить счет практически в p раз, поскольку обмен между ветвями чрезвычайно мал [7].

Будем различать левую и правую текущие ветви. Номера элементов, просматриваемых левой (правой) текущей ветвью, образуют возрастающую (убывающую) последовательность. Для левой (правой) текущей ветви нужно задать верхнюю (нижнюю) границу, до которой она движется. Таким образом, две ветви из p текущих ветвей могут либо двигаться друг другу навстречу, либо расходиться, либо двигаться в одном направлении (влево или вправо). Отметим, что в любой части дерева перебора можно сконцентрировать несколько текущих ветвей.

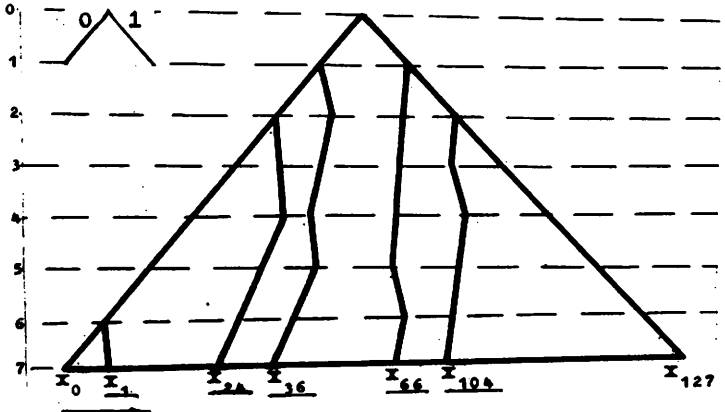


Рис. 2

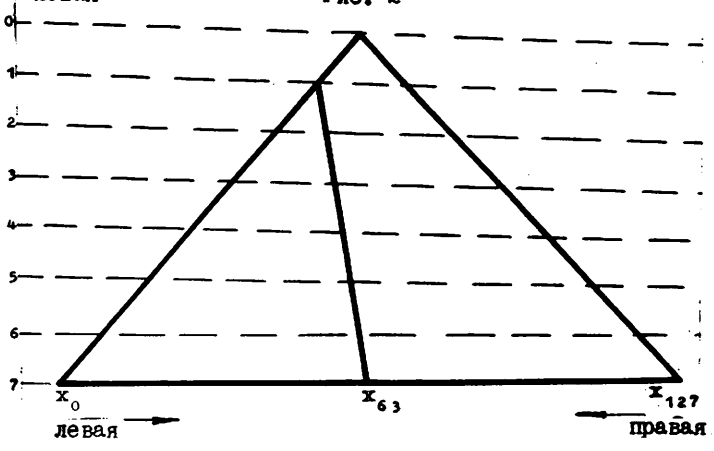


Рис. 3

ПРИМЕР. Найти все допустимые решения уравнения $1x_1+2x_2+3x_3 + 4x_4 + 5x_5 + 6x_6 + 7x_7 = 7$, $x_j = 0,1$; $j = \overline{1,7}$.

Поэлементный перебор требует подстановки в уравнение всех $2^7 = 128$ бинарных векторов и отбора тех из них, которые удовлетворяют ему. Ниже также осуществляется перебор всех 128 бинарных векторов. Однако ненужные элементы исключаются не по одному, а целыми множествами. Это - самое главное!

На рис.2 и 3 допустимые решения подчеркнуты, а исключаемые множества не подчеркнуты. Вектор $x_i$ представляет собой бинарный вектор $(\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{i7})$, где $i = \sum_{j=1}^{7} \alpha_{i,j} 2^{7-j}$. Так, например, $x_{23} = (0,0,1,0,1,1,1)$; $x_{24} = (0,0,1,1,0,0,0)$. Таким образом, бинарный вектор $x_i$ есть двоичное разложение своего индекса $i$. Поэтому нет необходимости выписывать векторы $x_i$ полностью ($i = \overline{0,127}$).

Левая ветвь 1-ветвевого обхода дерева перебора дает следующую последовательность векторов: $x_0\ x_1\ x_2 \ldots x_{23}\ x_{24}\ x_{25} \ldots x_{35}\ x_{36} \ldots x_{65}\ x_{66}\ x_{67} \ldots x_{103}\ x_{104}\ x_{105} \ldots x_{127}$.

На рис.2 и 3 изображены 1-ветвевой (левая) и 2-ветвевой (левая и правая) обходы дерева перебора.

Множество допустимых решений рассматриваемого уравнения состоит из векторов $x_1 = (0,0,0,0,0,0,1)$, $x_{24} = (0,0,1,1,0,0,0)$, $x_{36} = (0,1,0,0,1,0,0)$, $x_{66} = (1,0,0,0,0,1,0)$, $x_{104} = (1,1,0,1,0,0,0)$.

## 8. Рекуррентные соотношения

Рассмотрим задачу о ранце

$$\max\{\sum_{j=1}^{n} c_j x_j \mid \sum_{j=1}^{n} a_j x_j \le b,\ x_j \ge 0 \wedge x_j \in Z,\ j = \overline{1,n}\}.$$

Считаем, что заданные величины $a_j$, $b$ и $c_j$ удовлетворяют условиям $a_k \ne a_1$, $k \ne 1$, $a_j > 0 \wedge a_j \in Z$, $b \ge 0 \wedge b \in Z$, $c_j \ge 0$ ($j = \overline{1,n}$).

Для целочисленного аргумента $y = \overline{0,b}$ определяем функции

$$\phi_k(y) = \max\{\sum_{j=1}^{k} c_j x_j \mid \sum_{j=1}^{k} a_j x_j \le y,\ x_j \ge 0 \wedge x_j \in Z,$$

$$j = \overline{1,k}\},\quad k = \overline{1,n}.$$

Очевидно, $\phi_k(0) = 0$, $k = \overline{1,n}$, тогда $\phi_1(y) = c_1 [y/a_1]$, а для $k = \overline{2,n}$ имеют место рекуррентные соотношения $\phi_k(y) = \max(\phi_{k-1}(y), \phi_k(y-a_k) + c_k)$. Считаем, что $\phi_k(y) = -\infty$ для $y < 0$, $k = \overline{2,n}$.

В рекуррентном соотношении используется тот факт, что в оптимальном решении либо $x_k = 0$, либо $x_k \geq 1$. Для записи того, какой именно случай имеет место, вводим функции $i_k(y)$, которые определяются соотношениями

$$i_1(y) = \begin{cases} 0, & [y/a_1] = 0, \\ 1, & [y/a_1] \geq 1, \end{cases}$$

$$i_k(y) = \begin{cases} k, & \phi_{k-1}(y) \leq \phi_k(y-a_k) + c_k, \\ i_{k-1}(y), & \phi_{k-1}(y) > \phi_k(y-a_k) + c_k \end{cases} \quad (y=\overline{0,b};\ k=\overline{2,n}).$$

При помощи функций $i_k(y)$ вычисляется оптимальное решение для фиксированных k и y. Пусть $j_1 = i_k(y)$. Рассмотрим индексы $j_2 = i_k(y - a_{j_1})$, $j_3 = i_k(y - a_{j_1} - a_{j_2}), \ldots, 0 = i_k(y - a_{j_1} - a_{j_2} - a_{j_3} - \ldots)$. Сколько раз индекс j появляется в ряду $j_1, j_2, j_3, \ldots, 0$, таково положительное значение переменной $x_j$ в оптимальном решении, а остальные переменные равны нулю.

Отметим, что на шаге 1 нет необходимости хранить информацию о шагах $k \leq l-2$. Это простое замечание позволяет значительно уменьшить память, необходимую для вычисления функций $\phi_n(y)$, $i_n(y)$.

Теперь покажем, как на p параллельно работающих процессорах можно выполнить описанный алгоритм решения задачи о ранце.

Разбиваем множество $S = \{1,2,\ldots,n\}$ на p подмножеств $S_l = \{(l-1)[n/p], (l-1)[n/p] + 1, \ldots, l[n/p]\}$, $l=\overline{1,p}$ (для простоты считаем, что n делится на p). Это разбиение порождает разбиение строк a и c на p частей. Для каждого подмножества $S_l$ вычисляем функции $\phi_{S_l}(y)$, $i_{S_l}(y)$ от целочисленного аргумента $y=\overline{0,b}$ $(l=\overline{1,p})$.

Решаем новую задачу

$$\max\Big\{ \sum_{l=1}^{p} \phi_{S_l}(y_l) \Big| \sum_{l=1}^{p} (y_l) \leq b,\ y_l \geq 0 \wedge y_l \in Z,\ l = \overline{1,p} \Big\}.$$

Эта задача решается при помощи рекуррентных соотношений

$$\bar{\phi}_k(y) = \max_{s=0}^{y} (\bar{\phi}_{k-1}(s) + \phi_{S_k}(y-s)),\ \bar{y}_k(y) = \bar{s},$$

где $\bar{s}$ – наименьшее значение, для которого достигается максимум

($y = \overline{0,b}$; $k = \overline{1,p}$). В этом случае счет ускоряется примерно в $p$ раз. В табл. I и 2 отражен процесс решения одной и той же задачи о ранце соответственно на одном и двух процессорах.

ПРИМЕР. Максимизировать $1x_1 + 3x_2 + 5x_3 + 9x_4$ при условиях $2x_1 + 3x_2 + 4x_3 + 7x_4 \le 10$, $x_j \ge 0$, $x_j \in Z$ $(j=\overline{1,4})$.

Таблица I

| $\phi_k, i_k$ | y | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | I | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | I0 |
| $\phi_1(y)$ | 0 | I | I | 2 | 2 | 3 | 3 | 4 | 4 | 5 |
| $i_1(y)$ | 0 | I | I | I | I | I | I | I | I | I |
| $\phi_2(y)$ | 0 | I | 3 | 3 | 4 | 6 | 6 | 7 | 9 | 9 |
| $i_2(y)$ | 0 | I | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| $\phi_3(y)$ | 0 | I | 3 | 5 | 5 | 6 | 8 | I0 | I0 | II |
| $i_3(y)$ | 0 | I | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| $\phi_4(y)$ | 0 | I | 3 | 5 | 5 | 6 | 9 | I0 | I0 | I2 |
| $i_4(y)$ | 0 | I | 2 | 3 | 3 | 3 | 4 | 3 | 4 | 4 |

Таблица 2

| $\phi_{S_k}, i_{S_k}$ | y | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | I | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | I0 |
| $\phi_1(y)$ | 0 | I | I | 2 | 2 | 3 | 3 | 4 | 4 | 5 |
| $i_1(y)$ | 0 | I | I | I | I | I | I | I | I | I |
| $\phi_{1,3}(y)$ | 0 | I | I | 5 | 5 | 6 | 6 | I0 | I0 | II |
| $i_{1,3}(y)$ | 0 | I | I | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| $\phi_2(y)$ | 0 | 0 | 3 | 3 | 3 | 6 | 6 | 6 | 9 | 9 |
| $i_2(y)$ | 0 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| $\phi_{2,4}(y)$ | 0 | 0 | 3 | 3 | 3 | 6 | 9 | 9 | 9 | I2 |
| $i_{2,4}(y)$ | 0 | 0 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 4 |
| $\phi_{1,3;2,4}(y)$ | 0 | I | 3 | 5 | 5 | 6 | 9 | I0 | I0 | I2 |
| $y_{1,3;2,4}(y)$ | 0 | 2 | 0 | 4 | 4 | 0 | 0 | 8 | 2 | 0 |
| $i_{1,3;2,4}(y)$ | 0 | I | 2 | 3 | 3 | 2 | 4 | 3 | 4 | 4 |

Функции $i_4(y)$ и $i_{1,3;2,4}(y)$ отличаются для значения $y=6$. Это объясняется тем, что для этого значения $y$ существует два оптимальных решения. Из табл. I и 2 видно, что для рассматриваемой конкретной задачи о ранце $\phi_4(10) = \phi_{1,3;2,4}(10) = 12$, $i_4(10) = i_{1,3;2,4}(10) = 4$, $i_4(3) = i_{1,3;2,4}(3) = 2$.

Таким образом, оптимальное решение есть $x_2 = 1$, $x_4 = 1$, а оптимальное значение целевой функции равно I2.

## Литература


I. ЕВРЕИНОВ Э.В., ХОРОШЕВСКИЙ В.Г. Однородные вычислительные системы. - Новосибирск: Наука, 1978. - 316 с.

2. МИРЕНКОВ Н.Н. Параллельные алгоритмы для решения задач на однородных вычислительных системах. - Вычислительные системы, вып. 57, 1973, с. 3-32.

3. ХОХЛЮК В.И. Процедура один для нахождения наибольшего общего делителя и коэффициентов разложения. (Ин-т математики СО АН СССР, март, 1978). - 12 с. - Алгоритмы и программы. Информ. бюлл., 1979, № 1, с. 20.

4. ХОХЛЮК В.И., ЯСЕНЕВ М.В. Процедура для нахождения всех крайних лучей конуса. (Ин-т математи СО АН СССР, март, 1978). - 15 с. - Алгоритмы и программы. Информ. бюлл., 1979, № 1, с. 21.

5. ХОХЛЮК В.И., ЯСЕНЕВ М.В. Процедура для нахождения всех вершин и неограниченных направлений многогранника. (Ин-т математики СО АН СССР, май, 1978). - 20 с. - Алгоритмы и программы. Информ. бюлл., 1979, № 2, с. 21.

6. ХОХЛЮК В.И. Задачи целочисленной оптимизации (преобразования). - Новосибирск: НГУ, 1979. - 92 с.

7. ХОХЛЮК В.И. Задачи целочисленной оптимизации (алгоритмы). - Новосибирск : НГУ, 1980. - 92 с.

8. MANAS M., NEDOMA J. Finding All Vertices of a Convex Polyhedron. - Numer.Math., 1968, Bd 12, N 3, S.226-229.



Поступила в ред.-изд.отд.
16 июля 1980 года