$L_{\underset{W}{\sim}}$ сообщение вида (α, m_1), где m_1 — номер уровня вершины α, а для каждого листа β уровня $j < n$ — сообщения (β, m_2), $m_2 = \overline{j+1, n}$ (заметим, что в итоге не будет дублирующих сообщений).

```
1. процедура СООБЩЕНИЕ (y, r, x_{k_l}, k_l, τ);
2. пусть a_1a_2 ... a_r ⇄ y, b_1b_2 ... b_{k_l} ⇄ x_{k_l};
3. пусть τ_1τ_2 ... τ_n ⇄ τ;
4. j := 0;
5. для i = 1 на 1 пока a_i = b_i цикл;
6.    j := i;
   конец-цикл;
7. для i = j + 1 на 1 до r — 1 цикл;
8.    если τ_i = 0 то
9.       ПЕЧАТЬ-СООБЩЕНИЯ (a_1a_2 ... a_i, i);
      конец-если;
   конец-цикл;
   конец-процедура;
```

Рис. 2

Кроме того, в каждый данный момент работы алгоритма допускается обозревать по одному элементу из каждого V_i. Алгоритм должен использовать для всех таких задач с фиксированным n ограниченный объем оперативной памяти, не зависящий от мощности V_i, $i = \overline{1, n}$.

2. Алгоритм решения задачи приведен на рис. 1. Для получения очередного элемента методом последовательного доступа используется процедура СЛЕДУЮЩИЙ (V_i, x_i, p), которая при очередном выполнении с аргументом V_i размещает в x_i очередной элемент множества V_i; если $x_i = \omega_i$, то p увеличивается на 1, в противном случае — остается без изменения. Вызов процедуры после достижения элемента ω_i не допускается.

Процедура ОБРАБОТАТЬ (A, π) производит некоторую обработку полной цепи A.

Для поиска фиктивных вершин и печати о них сообщения используется процедура СООБЩЕНИЕ $(y, r, x_{k_l}, k_l, \tau)$ (рис. 2). Непосредственно печать сообщения в требуемом виде осуществляет процедура ПЕЧАТЬ-СООБЩЕНИЯ (α, i); здесь α — вершина, а i — уровень этой вершины.

ЛИТЕРАТУРА

1. Гендель Е. Г., Левин Н. А. Оптимизация технологии обработки информации в АСУ.— М., 1977, с. 232.
2. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов.— М., 1979, с. 536.
3. Скорняков Л. А. Элементы теории структур.— М., 1970, с. 148.

Поступила в редакцию 29.06.81.

Кафедра общего программирования

УДК 681.3.06.

П. В. ГЛЯКОВ

АЛГОРИТМ ЗАДАЧИ О РАЗБИЕНИИ И ЕГО ИССЛЕДОВАНИЕ НА ЭВМ

Рассмотрим задачу о разбиении. Дано множество $I = \{1, 2, \ldots, n\}$. Каждому элементу этого множества $i \in I$ поставлен в соответствие целый положительный вес $p_i \in P = \{p_1, p_2, \ldots, p_n\}$. Требуется разбить I на два подмножества I_1, $I_2 \subset I$, чтобы $I_1 \cap I_2 = \varnothing$, $I_1 \cup I_2 = I$ и $\sum\limits_{i \in I_1} p_i = \sum\limits_{i \in I_2} p_i$. Эта задача может быть записана как задача ЦЛП с булевыми переменными в следующем виде:

$$z = \sum_{i \in I} p_i x_i \to \max, \qquad (1)$$

$$\sum_{i \in I} p_i x_i \leqslant s, \; x_i \in \{0, 1\}, \; i \in I,$$

где $s = \frac{1}{2} \sum_{i \in I} p_i$, $p_i \geqslant 0$ и целые. Очевидно, что, если $z = s$, исходная задача о разбиении имеет решение, и подмножества I_1, I_2 можно выбирать следующим образом: $I_1 = \{i \in I \mid x_i = 1\}$, $I_2 = I \setminus I_1$.

Данная работа представляет собой попытку довести границу памяти методов динамического программирования $O(s)$ (см. [1, 2]) до s двоичных разрядов, не ухудшая при этом быстродействия. Это позволит решать задачу (1) со значением s, в 30—60 раз превышающим возможности методов динамического программирования. Показывается, что при $n < 1500$ предлагаемый алгоритм затрачивает не больше времени, чем методы динамического программирования. Заметим, что данное значение переменной n является пределом возможностей методов динамического программирования для ЭВМ типа ЕС с объемом памяти 512К байтов.

Для решения задачи (1) предлагается точный алгоритм, основанный на отображении суммарных значений весов всех подмножеств $I_k \subseteq I$, таких что $\sum_{i \in I_k} p_i \leqslant \frac{1}{2} \sum_{i \in I} p_i$, на двоичный вектор $T = (t_0, t_1, \ldots, t_s)$, где $s = \frac{1}{2} \sum_{i \in I} p_i$. Значения элементов вектора T определяется следующим образом:

$$t_j = \begin{cases} 1, & \text{если } \exists I_k \subseteq I, \text{ что } \sum_{i \in I_k} p_i = j, \\ 0, & \text{в противном случае,} \end{cases}$$

$j \in \{0, 1, \ldots, s\}$. Для выполнения такого отображения требуется $O(n)$ операций логического сдвига и логического сложения над $(s+1)$-мерными векторами. Более подробно с данным отображением можно познакомиться в работе [3].

Рассмотрим ряд свойств линейного уравнения с булевыми переменными вида

$$\sum_{i \in I} p_i x_i = s, \qquad (2)$$

где $I = \{1, 2, \ldots, n\}$, $P = \{p_1, p_2, \ldots, p_n\}$, $p_i > 0$ и целые, $x_i \in \{0, 1\}$, $i \in I$. В дальнейшем нам понадобится множество $Q = \{q_1, q_2, \ldots, q_n\}$, элементы которого натуральные числа, представляющие собой кратности вхождения элементов $p_i \in P$ в последовательность $\mathrm{P} = p_1, p_2, \ldots, p_n$. Кратность $q_i \in Q$ определим следующим образом: $q_i = |R|$, где $R = \{j \mid j \in \{1, 2, \ldots, i\}, p_j = p_i\}$. Далее потребуем, чтобы элементы последовательности P были упорядочены по невозрастанию, т. е. чтобы для любых $i < j$ выполнялось $p_i \geqslant p_j$.

*S*1. Если $\exists k \in I$, что уравнение $\sum_{i \in I \setminus \{k\}} p_i x_i = s$ не имеет решения и уравнение (2) имеет решение, то в решении уравнения (2) $x_i = 1$ для $\forall i \in \{k - q_k + 1, \ldots, k\}$.

*S*2. Если $\sum_{i \in I} p_i = s$, то в решении уравнения (2) $x_i = 1$ для $\forall i \in I$.

*S*3. Если $\exists k \in I$, что $\sum_{i \in I \setminus \{k\}} p_i = s$, то $x_i = 1$ для $\forall i \in I \setminus \{k\}$, $x_k = 0$ является решением уравнения (2).

*S*4. Если $\exists k \in I$, что уравнение $\sum_{i \in I} p_i x_i = s - p_k$ не имеет решения, а

уравнение (2) имеет решения, то в решении уравнения (2) $x_i = 0$ для $\forall i \in \{k - q_k + 1, \ldots, k\}$.

$S5$. Если $\exists k \in I$, что уравнение $\sum_{i \in I} p_i x_i = s - q_k p_k$ имеет решение, уравнение $\sum_{i \in I} p_i x_i = s - (q_k + 1) p_k$ не имеет решения и уравнение (2) имеет решение, то в решении уравнения (2) $x_i = 0$ для $\forall i \in \{k - q_k + 1, \ldots, k\}$,

$S6$. Если $\exists k \in I$, что уравнение $\sum_{i \in I} p_i x_i = s - q_k p_k$ имеет решение, $s < (q_k + 1) p_k$ и уравнение (2) имеет решение, то в решении уравнения (2) $x_i = 1$ для $\forall i \in \{k - q_k + 1, \ldots, k\}$.

Отметим, что приведенные свойства остаются справедливыми, если последовательность P неупорядочена. Но упорядоченность P будет использоваться для более быстрого решения задачи (1), так как она позволяет сократить количество просматриваемых элементов $p_i \in P$. Аналогично вычисление элементов множества Q, вообще говоря, является необязательным, и даже если известно, что равные веса отсутствуют или их «мало», то вычислять элементы $q_i \in Q$ нет необходимости. Но если равные веса присутствуют, а практически это часто бывает, то использование множества Q позволяет значительно улучшить быстродействие предлагаемого алгоритма.

Основная процедура для решения задачи (1) основывается на свойстве $S1$. Она всегда позволяет понизить размерность задачи. Остальные свойства $S2$—$S5$ проявляются случайным образом и часто оказываются полезными.

Перед выполнением алгоритма будем считать, что $R = \varnothing$, $n > 0$, $p_i > 0$ для $\forall\ i \in I$. Предполагается, что последовательность весов P упорядочена по невозрастанию и для каждого веса p_i вычислена его кратность q_i. Результатом работы алгоритма является множество $R \subseteq I$, такое что $X = (x_i = 1\ \forall i \in R,\ x_i = 0\ \forall i \in I \setminus R)$ есть решение задачи (1). Значение переменной z будет равно максимальному значению целевой функции задачи (1).

Алгоритм A

$A1$. [Проверить емкость рюкзака.] Если $\sum_{i \in I} p_i < s$, то $z = \sum_{i \in I} p_i$, $R = I$, перейти к $A16$.

$A2$. [Проверить свойство $S2$.] Если $\sum_{i \in I} p_i = s$, то $R = R \cup \{i \mid i \in I,\ p \neq 0\}$, перейти к $A16$.

$A3$. [Проверить свойство $S3$.] Если $\exists k \in I$, что $p_k \neq 0$, $\sum_{i \in I \setminus \{k\}} p_i = s$, то $R = R \cup \{i \mid i \in I \setminus \{k\},\ p_i \neq 0\}$, перейти к $A16$.

$A4$. [Освободить рюкзак.] $m = 1$; $t_0 = 1$; $t_i = 0$, $i = \overline{1, s}$.

$A5$. [Проверить предмет.] Если $p_m = 0$, перейти к $A8$. Если $p_m \leqslant s$, то перейти к $A6$, иначе $p_m = 0$, перейти к $A8$.

$A6$. [Проверить группу кратности.] Если $q_m \neq 1$ перейти к $A7$. Если $\exists k \in \{i \mid i \in I,\ p_i = p_m\}$, что $s - (k - m + 1) p_m \geqslant 0$, $t_{s-(k-m+1)p_m} = 1$, то перейти к $A10$.

$A7$. [Поместить предмет в рюкзак.] $T' = (T) p_m$, $T = T \oplus T'$.

$A8$. [Все ли предметы рассмотрены?] Если $m < n$, то $m = m + 1$, перейти к $A5$.

$A9$. [Рюкзак плотно не упаковывается.] Выполнить $s = s - 1$, пока t_s не станет равным 1. Положить $z = s$. Если $s = 0$ перейти к $A16$, иначе перейти к $A12$.

$A10$. [Запомнить предметы.] $R = R \cup \{i \mid i \in \{k - q_k + 1, \ldots, k\}\}$, $s = s - q_k p_k$.

Если $s=0$ перейти к $A16$, иначе $p_i=0$ для $\forall\, i\in\{k-q_k+1, \ldots, k\}$.

$A11$. [Понизить размерность задачи.] $n=m-1$.

$A12$. [Проверить свойство $S4$.] Положить $p_i=0$ для $\forall\, i\in I$, что $(p_i<s, p_i\neq 0, t_{s-p_i}=0) \vee (p_i>s)$.

$A13$. [Проверить свойство $S5$.] Если $\exists\, p_k$, $p_k\neq 0$, $k\in I$, что $s>(q_k+1)p_k$, $t_{s-q_kp_k}=1$, $t_{s-(q_k+1)p_k}=0$, то выполнить $A10$, перейти к $A2$.

$A14$. [Проверить свойство $S6$.] Если $\exists\, p_k$, $p_k\neq 0$, $k\in I$, что $q_kp_k\leqslant s<(q_k+1)p_k$, $t_{s-q_kp_k}=1$, то выполнить $A10$.

$A15$. [Цикл.] Перейти к $A2$.

$A16$. [Завершить алгоритм.]

Здесь на шаге $A7$ используются следующие обозначения: $(T)p_m$ обозначает логический сдвиг двоичного вектора T на величину p_m вправо; $T\oplus T'$ — логическое сложение двух двоичных векторов T и T'. Хотя при этих обозначениях используются два двоичных вектора T и T', в действительности же необходим только исходный вектор T и дополнительно поле длиной 256 байтов. Отметим, что при выполнении алгоритма изменяют свои значения переменные n, s и p_i, $i\in I$. Переменная n в каждый момент времени указывает количество просматриваемых весов $p_i(i\in I)$.

Трудоемкость алгоритма A будет вычислять относительно основной процедуры $S1$, которая реализуется шагами $A4$—$A11$. Влияние свойств $S2$—$S6$ на эффективность алгоритма аналитически трудно учесть из-за их вероятностной природы, поэтому при вычислении трудоемкости они не будут учитываться. Сами же свойства $S2$—$S6$ имеют сложность $O(n)$ и, будучи выполнены в цикле, потребуют $O(n^2)$ времени.

Найдем количество логических операций сложения и сдвига при k-ом выполнении процедуры $S1$. Легко заметить, что после каждого выполнения процедуры $S1$ величина s понижается в среднем на величину $\frac{s}{n}+\frac{s}{m}$, где n — количество весов в $\mathbf{P}$, m — количество различных весов в $\mathbf{P}$. Переменная n будет уменьшаться в среднем на величину $\frac{n}{m}$. Отсюда получаем, что при k-ом выполнении процедуры $S1$ будем иметь следующее количество обработанных битовых разрядов

$$B_k=\left(S-(k-1)\left(\frac{s}{n}+\frac{s}{m}\right)\right)\left(n-(k-1)\frac{n}{m}\right).$$

Тогда алгоритм A будет обрабатывать в среднем

$$B=\sum_{k=1}^{m/2}B_k=\frac{s}{24\cdot m}(7nm^2-2m^3+9nm+3m^2+2n+2m) \qquad (3)$$

битовых разрядов.

Шаг $A7$ в целях повышения эффективности выполняется на Ассемблере. Используются 3 типа команд Ассемблера в зависимости от значения величины p_i, на которую сдвигается вектор T.

1. Если $\mathrm{mod}_8(p_i)=0$, то для сдвига используется команда логического сложения CLC, позволяющая обрабатывать до $C_1=2048$ битовых разрядов за одно применение.

2. Если $\mathrm{mod}_4(p_i)=0$ и $\mathrm{mod}_8(p_i)\neq 0$, то используется команда пересылки со сдвигом MVO, которая позволяет обрабатывать до $C_2=120$ битовых разрядов.

3. В остальных случаях используется команда двойного сдвига SRDL, обрабатывающая до $C_3=56$ битовых разрядов.

Вычислим математическое ожидание обработанных битовых разрядов за одно применение «средней» команды.

$$M=\sum_{i=1}^{3}p(C_i)C_i=\frac{1}{8}2048+\frac{1}{8}120+\frac{3}{4}56=313.$$

Рассмотрим наихудший случай, когда одинаковые веса отсутствуют или их «мало», т. е. когда $m=n$. Тогда формула (3) принимает вид

$$B = \frac{s}{24}(5n^2 + 12n + 4).$$

В этом случае алгоритм A затрачивает следующее количество логических команд:

$$\text{Эф} = \frac{s}{24 \cdot 313}(5n^2 + 12n + 4). \qquad (4)$$

Сравним полученную оценку (4) с оценкой методов динамического программирования в предположении, что они, имея сложность $O(sn)$, затрачивают по крайней мере $s \cdot n$ команд. Получим, что при $n < 1500$ алгоритм A выполняется быстрее.

Описанный алгоритм запрограммирован на языке Ассемблера. Программа просчитывалась на ЭВМ ЕС-1022, имеющей быстродействие 80 тысяч операций в секунду. Для каждого значения емкости генерировались случайные числа в интервалах [1, 99] и [1, 999]. Результаты машинных экспериментов приведены в таблице, где время представляет собой среднее значение из 10 сгенерированных задач для каждого значения емкости рюкзака.

Емкость рюкзака s	Интервал [1,99]		Интервал [1,999]	
	Количество переменных n	Время, с	Количество переменных n	Время, с
50000	2000	56,81	200	50,37
60000	2400	122,42	240	109,97
70000	2800	193,61	280	173,83
80000	3200	262,73	320	244,46
90000	3600	341,07	360	321,94
100000	4000	423,26	400	409,70

Время, необходимое для упорядочения последовательности P и вычисления кратности q_i, не включается. Приведенное время есть время работы центрального процессора ЭВМ.

В заключение остановимся на одном важном подходе, позволяющем решать задачи (1) больших размерностей с достаточно большими значениями весов p_i. Для этой цели предлагается первоначально загрузить некоторую часть емкости рюкзака s_1 произвольно выбранными «большими» предметами, а затем решать задачу (1) для оставшейся емкости $s_2 = s - s_1$. Величина s_1 выбирается, исходя из наличия доступной памяти, времени и качества искомого решения. Использование алгоритма A в этом случае имеет преимущество по сравнению с методами динамического программирования, так как в задаче (1), полученной после приведения, емкость рюкзака может быть значительно больше, чем это позволяют методы динамического программирования, а именно, $s_2 = 2 \cdot 10^6$.

ЛИТЕРАТУРА

1. Ковалев М. М. Дискретная оптимизация (целочисленное программирование).— Минск, 1977.

2. Lawler E. L.— Mathematics of Operations Research, 1979, v. 4, № 4, p. 339.

3. Гляков П. В., Глякова Л. А.— В сб.: Математическое обеспечение ЭВМ и АСУ.— Минск, 1981, с. 62.

Поступила в редакцию
29.06.81.

Кафедра общего программирования