$$\begin{cases}[\alpha + p_i(\beta - \alpha)]\, a_i' d + a_i' \overline{Db} - b_i \leqslant 0, \\ [\alpha + (1 - p_i)(\beta - \alpha)] a_i' d + a_i' \overline{Db} - b_i \leqslant 0, \quad i = \overline{2,\ m}. \end{cases}$$

Итак, в рассматриваемом случае, задача (1)—(3) сводится к задаче обычного линейного программирования.

Рассмотрим следующий

Пример.

$$M(x_1 + x_2) \to \min, \tag{10}$$

$$P\{2x_1 - 4x_2 \leqslant b_1\} \geqslant 0{,}7, \tag{11}$$

$$P\{-3x_1 + x_2 \leqslant 6\} \geqslant 0{,}6, \tag{12}$$

где случайная величина b_1 распределена равномерно в интервале $[-2,2]$, а решающее правило задано в виде (3).

В соответствии с изложенной теорией, решение задачи (10)—(12) сводится к решению следующей задачи линейного программирования:

$$d_{21} + d_{22} \to \min,$$

$$\begin{cases} 0{,}4d_{11} - 0{,}8d_{21} + 3d_{12} - 6d_{22} \leqslant 0{,}2, \\ -0{,}4d_{11} + 0{,}8d_{21} + 3d_{12} - 6d_{22} \leqslant -0{,}2, \\ -0{,}6d_{11} + 0{,}2d_{21} - 9d_{12} + 3d_{22} \leqslant 3, \\ 0{,}6d_{11} - 0{,}2d_{21} - 9d_{12} + 3d_{22} \leqslant 3, \end{cases}$$

которая имеет следующее решение: $d_{11}^0 = -0{,}1$; $d_{12}^0 = -0{,}4$; $d_{21}^0 = -0{,}3$; $d_{22}^0 = -0{,}2$. Таким образом, решение задачи (10)—(12) имеет вид: $x_1^0 = -0{,}1 \cdot b_1 - 2{,}4$; $x_2^0 = -0{,}3 \cdot b_1 - 1{,}2$.

ЛИТЕРАТУРА

1. Юдин Д. Б. Математические методы управления в условиях неполной информации.— М., 1974.

2. Ермольев Ю. М. Методы стохастического программирования.— М., 1976.

Поступила в редакцию 10.06.81.

Кафедра теории вероятностей и математической статистики

УДК 681.3.06

Г. А. ДРОБУШЕВИЧ, И. В. КОМАРОВСКИЙ

ОБ ОДНОЙ МОДЕЛИ ЗАДАЧИ СОВМЕСТНОЙ ОБРАБОТКИ ФАЙЛОВ

Под задачей совместной обработки файлов будем понимать управление передвижением файлов с последовательной организацией, используемых программой. В [1] предлагаются методы проектирования программ совместной обработки $n (n \geqslant 2)$ одинаково упорядоченных файлов с оценкой сложности проектирования $O(2^n)$. Это фактически означает, что в одном программном модуле необходимо ограничиваться двумя-тремя входными файлами, что никак не способствует эффективности (по времени и памяти) решения задач АСУП.

В настоящей работе описывается математическая модель и приводится алгоритм решения достаточно широкого класса задач, а именно — задач совместной обработки упорядоченных файлов, множество значений ключей которых образует древовидную структуру соответствия записей. Этот класс охватывает большую часть задач АСУП, которые могут быть решены с использованием методов совместной обработки файлов. При наличии алгоритма решения задач этого класса сложность проектирования программ решения таких задач уже не зависит от числа входных файлов.

1. Файл **F** можно представить как множество записей $\{z\}$, а каждую запись — как слово в некотором алфавите V. Мы будем рассматривать

однородные файлы и, значит, длина записи в пределах файла будет одинаковой. Запись $z \in \mathbf{F}$ состоит из полей — непересекающихся подслов слова z; следовательно, можем записать $z = z_1 z_2 \ldots z_m$. Каждая запись в пределах файла имеет одно и то же число полей и длина i-ых полей в различных записях одинакова. Множество $\mathbf{p}_i$ i-ых подслов (полей) во всех записях файла называют реквизитом, а i-ое подслово некоторой записи z — значением реквизита $\mathbf{p}_i$ (обозначим его через $z(\mathbf{p}_i)$). Множество реквизитов $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m\}$ разделяют на два непустых подмножества: $\mathbf{R}$ — реквизитов-признаков и $\mathbf{Q}$ — реквизитов-оснований, $\mathbf{P} = \mathbf{R} \cup \mathbf{Q}$, $\mathbf{R} \cap \mathbf{Q} = \varnothing$. Само множество $\mathbf{R}$ называют ключом файла $\mathbf{F}$, а множество принадлежащих некоторой записи значений реквизитов-признаков из $\mathbf{R}$ — значением ключа.

Пусть имеется $n (n \geqslant 2)$ входных файлов $\mathbf{F}_i$, $i = \overline{1, n}$. Записи $z^i \in F_i$ и $z^j \in \mathbf{F}_j$ назовем корреспондирующими (соответствующими), если $\mathbf{R}_i \cap \mathbf{R}_j \neq \varnothing$ и $(\forall \mathbf{p} \in \mathbf{R}_i \cap \mathbf{R}_j)[z^i(\mathbf{p}) = z^j(\mathbf{p})]$. Если $\mathbf{R}_1 \subseteq \mathbf{R}_2 \subseteq \ldots \subseteq \mathbf{R}_n$, то на множестве записей файлов $\mathbf{F}_i$, $i = \overline{1, n}$, отношение корреспондирования определяет совокупность ориентированных деревьев, в вершинах которых находятся значения ключей файлов. Такие деревья будем называть КЗ *деревьями* (деревьями корреспондирующих записей).

Ниже рассматривается одна из моделей совместной обработки файлов и приводится алгоритм решения для случая, когда $\mathbf{R}_1 \subset \mathbf{R}_2 \subset \ldots \subset \mathbf{R}_n$. Отсюда уже нетрудно будет получить алгоритм решения, когда $\mathbf{R}_1 \subseteq \mathbf{R}_2 \subseteq \ldots \subseteq \mathbf{R}_n$.

Перейдем к описанию модели. Пусть задан конечный алфавит V. Тогда V^* — множество всех слов над V. В обозначениях слов из V^* нижний индекс будет указывать длину слова, т. е. если $\alpha_i \in V^*$, то $i = |\alpha_i|$. На V^* определим отношение $\leqslant^{\text{п}}$: для $\alpha, \beta \in V^*$ $\alpha \leqslant^{\text{п}} \beta$ тогда и только тогда, когда α — начальное подслово слова β, т. е. $\beta = \alpha\alpha'$. Ясно, что это отношение частичного порядка на V^*.

Нетрудно убедиться, что отношение $\leqslant^{\text{п}}$ удовлетворяет следующему условию: если $\alpha \leqslant^{\text{п}} \gamma$ и $\beta \leqslant^{\text{п}} \gamma$, то α и β сравнимы. Такое отношение будем называть отношением преддревесного порядка, а упорядоченную пару $L = \langle V^*, \leqslant^{\text{п}} \rangle$ — лесом (ориентированным). Если $T \subseteq V^*$, то $L_T = \langle T, \leqslant^{\text{п}}_T \rangle$ также будет лесом; здесь $\leqslant^{\text{п}}_T$ — ограничение отношения $\leqslant^{\text{п}}$ на T. Если в $\langle T, \leqslant^{\text{п}}_T \rangle$ для каждой пары элементов существует нижняя грань, то отношение $\leqslant^{\text{п}}_T$ будем называть древесным порядком, а L_T — деревом (ориентированным).

Терминологию, относящуюся к лесам и деревьям, будем использовать из [2].

Выберем произвольное подмножество $T \subseteq V^*$ и рассмотрим лес $L_T = \langle T, \leqslant^{\text{п}}_T \rangle$. Заметим, что частично упорядоченное множество T удовлетворяет условию минимальности [3]. Отсюда нетрудно показать, что лес L_T можно представить как совокупность деревьев, корнями которых являются минимальные элементы множества $\langle T, \leqslant^{\text{п}}_T \rangle$.

Зададим на V линейный порядок $\leqslant'$. Тогда естественным образом можно определить отношение лексикографического порядка $\leqslant^{\text{л}}$ на V^*.

В V^* выделим подмножества V_i', $i = \overline{1, n}$, каждое из которых содержит все слова из V^* длины i. Зафиксируем $V_i \subseteq V_i'$, $i \subseteq N = \{1, 2, \ldots, n\}$. Для построения алгоритма удобно в V выделить символ ω такой, что $(\forall a \in V)[a \neq \omega \to a <' \omega]$ и считать, что $\omega_i \rightleftarrows \omega^i \in V_i$, где $\omega^i = \underbrace{\omega \ldots \omega}_{i \text{ раз}}$, и $(\forall \alpha_i \subseteq V_i)[\alpha_i \neq \omega_i \to \omega \notin \alpha_i]$ (здесь и ниже знак $\rightleftarrows$ используется вместо слов «вводится в качестве обозначения»). Отсюда, очевидно, $\alpha_i <^{\text{л}} \omega_i$ для $\forall \alpha_i \in V_i \setminus \{\omega_i\}$.

Пусть $W = \bigcup_{i=1}^{n} V_i$. Тогда $L_W = \langle W, \leqslant^{\text{п}}_W \rangle$ — лес, который, как легко видеть, состоит из конечного числа деревьев. Это и есть КЗ деревья. Пол-

ной цепью леса L_W назовем путь $A = \langle \alpha_{k_1}, \alpha_{k_2} \ldots \alpha_{k_l} \rangle$ (для краткости будем обозначать $A = \alpha_{k_1} \alpha_{k_2} \ldots \alpha_{k_l}$) из корня дерева в лист. Вершины леса длины i отнесем к вершинам i-го уровня.

Возьмем $X_i \subseteq V_i'$, $i = \overline{1, n}$, $X = \bigcup_{i=1}^{n} X_i$. Тогда $L_X = \langle X, \leqslant_X^{\text{и}} \rangle$—лес. Через $\Omega(V^*; X_1, X_2, \ldots X_n)$ обозначим множество полных цепей леса L_X.

Пусть U_i — множество всех слов длины i, являющихся начальными подсловами некоторых слов из W. Рассмотрим лес $L_{\widetilde{W}} = \langle \widetilde{W}, \leqslant_{\widetilde{W}}^{\text{и}} \rangle$, где $\widetilde{W} = \bigcup_{i=1}^{n} (V_i \cup U_i)$. Каждая полная цепь этого леса содержит вершины всех уровней, не превосходящих уровня листа. Заметим, что листья лесов $L_{\widetilde{W}}$ и L_W совпадают, а каждая полная цепь леса $L_{\widetilde{W}}$ содержит все вершины полной цепи леса L_W. Вершины леса $L_{\widetilde{W}}$, отсутствующие в лесе L_W, назовем фиктивными.

К элементам из $\langle V_i, \leqslant^{\text{л}} \rangle$ определим последовательный метод доступа так: при первом обращении к V_i получаем наименьший элемент, а при i-ом ($i > 1$) — элемент непосредственно следующий за элементом, полученным при $(i-1)$-ом обращении; после получения наибольшего элемента следующее обращение к V_i недопустимо.

Теперь сформулируем нашу задачу.

Задача. Заданы множества $\langle V_i, \leqslant^{\text{л}} \rangle$, $i = \overline{1, n}$. Построить алгоритм, который, используя последовательный метод доступа, а) выделяет все полные цепи леса L_W; б) выдает для каждой фиктивной вершины α леса

Начало;

1. пусть $\pi \rightleftarrows \pi_1 \pi_2 \ldots \pi_n$, $\mu \rightleftarrows \mu_1 \mu_2 \ldots \mu_n$, $\tau \rightleftarrows \tau_1 \tau_2 \ldots \tau_n$—строки из n битов;
2. $\mu := 0^n$; $p := 0$; $s := 0$;
3. **для** $i \in N$ **цикл**; СЛЕДУЮЩИЙ (V_i, x_i, p); **конец-цикл**;
4. **пока** $p \neq n$ **цикл**;
5. пусть $\alpha_i \rightleftarrows \lambda(x_i)$, $i = \overline{1, n}$, где $\lambda(x_i)$ — содержимое x_i;
6. $A := \min \langle \Sigma, \leqslant^{\text{л}} \rangle$, где $\Sigma \rightleftarrows \Omega(V^*; \{\alpha_1\}, \{\alpha_2\}, \ldots, \{\alpha_n\})$;
7. пусть $x_{k_1} x_{k_2} \ldots x_{k_l} \rightleftarrows A$, $1 \leqslant k_1 < k_2 < \ldots < k_l \leqslant n$, $1 \leqslant l \leqslant n$;
8. пусть $b_1 b_2 \ldots b_{k_l} \rightleftarrows x_{k_l}$;
9. сформировать битовую строку π такую, что
   $(\pi_{k_1} = \pi_{k_2} = \ldots = \pi_{k_l} = 1) \,\&\, (\forall\, j \in N \setminus \{k_1, k_2, \ldots, k_l\})\, [\pi_j = 0]$;
10. **если** $\mu_{k_l} = 0$ **то**
11. **комментарий** распознана полная цепь дерева;
12. ОБРАБОТАТЬ (A, π);
13. **если** $s \neq 0$ **то**
14. СООБЩЕНИЕ $(y, r, x_{k_l}, k_l, \tau)$;

    иначе
15. $s := 1$;

    конец-если;
16. **для** $i = k_l + 1$ **на** 1 **до** n **цикл**;
17. ПЕЧАТЬ-СООБЩЕНИЯ (x_{k_l}, i);

    конец-цикл;
18. $r := k_l$; $\tau := \pi$; $y := x_{k_l}$; пусть $a_1 a_2 \ldots a_r \rightleftarrows y$;

    конец-если;
19. $\mu := \pi$ и затем в μ заменить последнюю 1 на 0;
20. СЛЕДУЮЩИЙ (V_{k_l}, x_{k_l}, p);

    конец-цикл;
21. **если** $s \neq 0$ **то** $k_l := n$; СООБЩЕНИЕ $(y, r, x_{k_l}, k_l, \tau)$;
22. **конец-если**;

конец;

Рис. 1

$L_{\underset{W}{\sim}}$ 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: