ИЗВЕСТИЯ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ

№ 5 (311) ТЕХНОЛОГИЯ ТЕКСТИЛЬНОЙ ПРОМЫШЛЕННОСТИ 2008

УДК 677.024: 519.15

СОВЕРШЕНСТВОВАНИЕ БАЗОВОГО АЛГОРИТМА МИНИМИЗАЦИИ КОЛИЧЕСТВА НЕОБХОДИМЫХ ПРОБОРОК ОСНОВ В РЕМИЗ

Г.И. БОРЗУНОВ

(Московский государственный текстильный университет им. А.Н. Косыгина)

Ранее в [1] было показано, что решение задачи минимизации количества необходимых перезаправок основ в ремиз представляет собой поиск минимального разбиения множества исходных проборок основ в ремиз, для каждого подмножества которого существует унифицированная схема проборки основ в ремиз, способная заменить все схемы, принадлежащие этому подмножеству разбиения, и удовлетворяющая технологическим ограничениям. При этом основным технологическим ограничением является максимально допустимое количество используемых ремизок, определяемое возможностями ткацкого станка, на котором предполагается выработка тканей. Поиск минимального разбиения основывается на конструктивном перечислении разбиений исходного множества, состоящего из n элементов, и проверки выполнения указанного ограничения для каждого варианта разбиения. Число таких вариантов равно соответствующему числу Белла (B_n) и, следовательно, быстро возрастает с возрастанием мощности разбиваемого множества: $B_{15}\approx10^9$, $B_{20}\approx5\cdot10^{13}$. Поэтому возможность решения указанной задачи за допустимый интервал времени существенно зависит от эффективности используемых алгоритмов конструктивного перечисления разбиений множеств. Базовые алгоритмы перечисления разбиений множеств, описанные Романовским И.В. [2] и Липским В. [3], не обеспечивают перечисления разбиений в порядке монотонного возрастания числа подмножеств в разбиениях. В табл. 1 приводятся результаты последовательной генерации всех возможных разбиений множества X, состоящего из 4 элементов, с помощью алгоритма А, описанного в книге [2].

Каждый вариант разбиения в табл. 1 представляется характеристическим вектором: pChi, имеющим 4 координаты взаимно однозначно соответствующие элементам множества X. Числовое значение каждой из этих координат определяет номер подмножества, которому принадлежит соответствующий элемент множества X. Рассмотрим пятый вариант разбиения множества X: pChi[]= (1, 1, 2, 3). Этот вектор определяет разбиение X на 3 подмножества: первый и второй элементы X принадлежат первому подмножеству, третий элемент X принадлежит второму подмножеству, а четвертый элемент X – третьему подмножеству. Шестой характеристический вектор определяет разбиения X на два подмножества: pChi[]= (1, 1, 2, 1). Такое немонотонное увеличение числа подмножеств в разбиениях приводит к необходимости анализа всех вариантов разбиения заданного множества. Ранее для устранения необходимости анализа всех вариантов разбиения заданного множества, был разработан рекурсивный алгоритм, который реализует перечисление разбиений множества X при монотонном возрастании числа подмножеств в разбиениях [4].

Таблица 1

№ п/п	Базовый алгоритм А [2]						Алгоритм Б, разработанный автором				
1	pPsi[]=	1	1	1	1		pPsi[]=	1	1	1	1
2	pChi[]=	1	1	1	1		pChi[]=	1	1	1	1
3	pPsi[]=	1	1	1	2		pPsi[]=	1	1	1	2
4	pChi[]=	1	1	1	2		pChi[]=	1	1	1	2
5	pPsi[]=	1	1	2	2		pPsi[]=	1	1	2	2
6	pChi[]=	1	1	2	1		pChi[]=	1	1	2	1
7	pPsi[]=	1	1	2	2		pPsi[]=	1	1	2	2
8	pChi[]=	1	1	2	2		pChi[]=	1	1	2	2
9	pPsi[]=	1	1	2	3		pPsi[]=	1	2	2	2
10	pChi[]=	1	1	2	3		pChi[]=	1	2	1	1
11	pPsi[]=	1	2	2	2		pPsi[]=	1	2	2	2
12	pChi[]=	1	2	1	1		pChi[]=	1	2	1	2
13	pPsi[]=	1	2	2	2		pPsi[]=	1	2	2	2
14	pChi[]=	1	2	1	2		pChi[]=	1	2	2	1
15	pPsi[]=	1	2	2	2		pPsi[]=	1	2	2	2
16	pChi[]=	1	2	2	1		pChi[]=	1	2	2	2
17	pPsi[]=	1	2	2	2		pPsi[]=	1	1	2	3
19	pChi[]=	1	2	2	2		pChi[]=	1	1	2	3
20	pPsi[]=	1	2	2	3		pPsi[]=	1	2	2	3
21	pChi[]=	1	2	1	3		pChi[]=	1	2	1	3
22	pPsi[]=	1	2	2	3		pPsi[]=	1	2	2	3
22	pChi[]=	1	2	2	3		pChi[]=	1	2	2	3
23	pPsi[]=	1	2	3	3		pPsi[]=	1	2	3	3
24	pChi[]=	1	2	3	1		pChi[]=	1	2	3	1
25	pPsi[]=	1	2	3	3		pPsi[]=	1	2	3	3
26	pChi[]=	1	2	3	2		pChi[]=	1	2	3	2
27	pPsi[]=	1	2	3	3		pPsi[]=	1	2	3	3
28	pChi[]=	1	2	3	3		pChi[]=	1	2	3	3
29	pPsi[]=	1	2	3	4		pPsi[]=	1	2	3	4
30	pChi[]=	1	2	3	4		pChi[]=	1	2	3	4

В данной статье описывается разработанный автором итеративный алгоритм Eq1_1, который также последовательно генерирует варианты разбиения множества X в порядке монотонного возрастания числа подмножеств, но по сравнению с алгоритмом, предложенным в работе [4], обеспечивает дополнительные возможности совершенствования поиска экстремальных разбиений. Ниже приводится псевдокод алгоритма Eq1_1.

1. Пусть n – число элементов в множестве X; np – текущее число подмножеств в разбиении, ic – счетчик числа рассмотренных разбиений; i,ii, – управляющие параметры циклов; k – рабочая переменная; pPsi – вектор спецификации разбиения; pChi – характеристический вектор разбиения.

2. Установить значения характеристического вектора pChi в соответствии с разбиением множества X, состоящим из единственного класса, и значения вектора спецификации этого разбиения pPsi: for(i=0;i<n;i++) Chi[i]=pPsi[i]=1.

3. Положить ic=1 и вывести первый вектор разбиения pChi.

4. Выполнить np=2.

5. Если (np<=n), то перейти к п. 6; иначе стоп – все разбиения получены.

6. Положить ii=np.

7. Установить начальные значения характеристических векторов pPsi и pChi в соответствии с текущим значением np:

for (i=n-1; i>0; i--) {if(ii>1){pChi[i] = =pPsi[i]=ii; ii--;}else pChi[i]=pPsi[i]=1;}.

8. Положить ic++ и вывести вектор разбиения pChi.

9. Положить i=n-1.

10. Если i>0, то к п. 11; иначе перейти к п. 22.

11. Положить i=n-1.

12. Если выполняется ((pPsi[i]==pPsi[i--1])&&(pPsi[i]<np), то перейти к п. 13; иначе – к п. 15.

13. Перейти к следующему варианту pPsi, соответствующему текущему значению np, и установить начальный вариант pChi: pPsi[i]++; pChi[i]=pPsi[i]; k=np; for(ii=n-1; ii>i; ii--) {pPsi[ii] =k; if(k>pPsi[i]) k--;} for(ii=1; ii<n; ii++) if(pPsi[ii]==pPsi[ii-1]) pChi[ii]=1.

14. Положить ic++, вывести вектор разбиения pChi и перейти к п.16.

15. Положить i--; если i>0, то перейти к п.12; иначе перейти к п.16.

16. Если i>0, то перейти к п.17; иначе перейти к п. 10.

17. Положить ii=n-1.

18. Если ii>0, то перейти к п.19; иначе перейти к п. 10.

19. Если (pChi[ii]+1<=pPsi[ii]), то перейти к п. 20; иначе - перейти к п. 21.

20. Положить pChi[ii]++; k=ii; k++; выполнить цикл while(k<n) {if(pPsi[k]==pPsi[k-1]) {pChi[k]=1; } k++;}; положить ii=n-1; ic++; вывести вектор разбиения pChi; перейти к п.18.

21. Положить ii-- и перейти к п. 18.

22. Положить np++ и перейти к п. 5.

Приведенные в табл. 1 результаты показывают, что предлагаемый алгоритм Б обеспечивает перечисление вариантов разбиений множества при монотонном возрастании числа подмножеств разбиений.

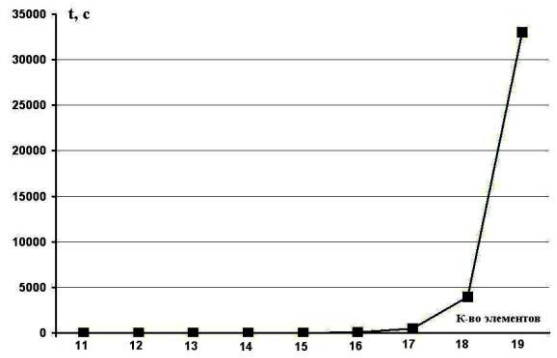

Рис. 1

На рис. 1 представлен график зависимости времени генерации всех разбиений множества от числа элементов в множестве. Расчеты были выполнены с использованием экспериментальной реализации алгоритма Eq1_1 на компьютере, оснащенном процессором Intel Core 2 Quad Q6600 EMT64 (2.40 ГГц). Этот вычислительный эксперимент показал, что для поиска экстремальных разбиений множеств, мощность которых превышает 19, необходимо использовать более мощные аппаратные средства, или вместо последовательного поиска применять модифицированный вариант бинарного поиска.

ВЫВОДЫ

1. Разработанный итеративный алгоритм конструктивного перечисления разбиений множеств (алгоритм Б в табл. 1) обеспечивает перечисление вариантов разбиений множества при монотонном возрастании числа подмножеств в разбиениях.

2. Этот алгоритм открывает новые возможности совершенствования поиска минимальных разбиений заданных множеств с использованием модификации метода деления отрезка пополам, что обеспечивает существенное уменьшение временной сложности задачи минимизации количества необходимых перезаправок основ в ремиз.

ЛИТЕРАТУРА

1. *Борзунов Г.И., Власов П.В.* Анализ и алгоритмы построения проборок основ при производстве семейства тканей с различными переплетениями // Изв. вузов. Технология текстильной промышленности. – 1979, №1. С. 42…45.

2. *Романовский И. В.* Алгоритмы решения экстремальных задач. – М.: Гл. ред. физ.-мат. лит-ры изд-ва «Наука»,1971.

3. *Липский В.* Комбинаторика для программистов. – М.: Мир, 1988.

4. *Борзунов Г. И., Пронин А. К.* Безопасность информационных технологий. – 2004, №4. С.58…60.

Рекомендована кафедрой информационных технологий и компьютерного дизайна. Поступила 03.02.08.