УДК 677.024: 519.15

ДВОИЧНЫЙ ПОИСК И ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ ПРИ МИНИМИЗАЦИИ КОЛИЧЕСТВА НЕОБХОДИМЫХ ПРОБОРОК ОСНОВ В РЕМИЗ

Г.И. БОРЗУНОВ

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

Ранее в [1] был описан разработанный автором итеративный алгоритм Eq1_1, который последовательно генерирует варианты разбиения множества в порядке монотонного возрастания числа подмножеств. В данной статье рассматривается усовершенствованный (на основе алгоритма Eq1_1) итеративный алгоритм Eq2_1. Алгоритм Eq2_1 также последовательно генерирует варианты разбиения множества в порядке монотонного возрастания числа подмножеств, но в отличие от алгоритма Eq1_1 для управления его работой используются два новых параметра: ns и nf. Значение параметра ns определяет начальное (минимальное) значение числа подмножеств в генерируемых разбиениях, а значение параметра nf – конечное (максимальное) значение числа подмножеств в генерируемых разбиениях. При равенстве значений этих параметров, например ns= nf=5, алгоритмом Eq2_1 генерируются все возможные разбиения, которые содержат в точности 5 подмножеств (табл. 1). В табл. 1 разбиения множества, состоящего из 6 элементов, представлены характеристическими векторами pChi[i], соответствующими векторам спецификации pPsi[i]. Структура векторов pChi[i] и pPsi[i] и их роль в генерации разбиений множеств описываются в [1].

Таблица 1

№ п/п	pPsi[i] –вектор спецификации						pChi[i] – характеристический вектор разбиения множества					
1	1	1	2	3	4	5	1	1	2	3	4	5
2	1	2	2	3	4	5	1	2	1	3	4	5
3	1	2	2	3	4	5	1	2	2	3	4	5
4	1	2	3	3	4	5	1	2	3	1	4	5
5	1	2	3	3	4	5	1	2	3	2	4	5
6	1	2	3	3	4	5	1	2	3	3	4	5
7	1	2	3	4	4	5	1	2	3	4	1	5
8	1	2	3	4	4	5	1	2	3	4	2	5
9	1	2	3	4	4	5	1	2	3	4	3	5
10	1	2	3	4	4	5	1	2	3	4	4	5
11	1	2	3	4	5	5	1	2	3	4	5	1
12	1	2	3	4	5	5	1	2	3	4	5	2
13	1	2	3	4	5	5	1	2	3	4	5	3
14	1	2	3	4	5	5	1	2	3	4	5	4
15	1	2	3	4	5	5	1	2	3	4	5	5

Таким образом, использование алгоритма Eq2_1 обеспечивает возможность генерации разбиений не подряд, а выборочно, что позволяет использовать при поиске минимальных разбиений схему двоичного поиска. Ниже приводится разработанный автором алгоритм двоичного поиска минимальных разбиений заданных множеств EQ3_1.

1. Пусть исходное множество X содержит n элементов. Положить ns=1, nf=n, P=(0,0,0,…,0), где P – характеристический вектор.

2. Положить np=nf. Используя алгоритм EQ2_1, выполнить поиск допустимого разбиения множества X на np частей. Если найден характеристический вектор допустимого разбиения pChi, то выполнить: {P= pChi; nf--;} иначе решения не существует, стоп.

3. Выполнить np= (nf + ns)/2. Здесь деление выполняется нацело, то есть с отбрасыванием остатка. Используя алгоритм EQ2_1, выполнить поиск допустимого разбиения множества X на np частей. Если найден характеристический вектор допустимого разбиения pChi, то выполнить: P=pChi; иначе перейти к п.5.

4. Положить nf = np; если ns < nf, то перейти к п. 3; иначе P – характеристический вектор минимального разбиения множества X, стоп,

5. Если ns < nf, то выполнить ns= np+1; иначе P – характеристический вектор минимального разбиения множества X, стоп,

6. Выполнить np= (nf + ns)/2. Деление выполняется нацело. Используя алгоритм EQ2_1, выполнить поиск допустимого разбиения множества X на np частей. Если найден характеристический вектор допустимого разбиения pChi, то выполнить: { P= pChi и перейти к п. 4;} иначе перейти к п. 5.

В табл. 2 приводятся результаты теоретического сравнения алгоритма EQ1_1 с ранее предложенными алгоритмами [2], [3] и алгоритма EQ3_1 с алгоритмом EQ1_1.

Таблица 2

Число элементов (n)	Ускорение алгоритма EQ1_1 по сравнению с алгоритмами [2], [3]		Ускорение алгоритма EQ3_1 по сравнению с EQ1_1	
	в худшем	в среднем	в худшем	в среднем
10	1	2	1,69	1,05
20	1	2	1,67	1,15
30	1	2	1,66	1,20
40	1	2	1,85	1,32
50	1	2	1,78	1,32

На основе алгоритма EQ2_1 впервые разработана вычислительная схема параллельного поиска минимального разбиения (алгоритм EQ4_1). Псевдокод алгоритма EQ4_1 приводится ниже. При описании параллельных операций используются обозначения, принятые в книге Дж. Макконелла "Анализ алгоритмов".

1. Пусть n – число элементов в исходном множестве X; p – число процессоров, доступных для выполнения поиска; fp[n] – вектор активных процессоров: nsp[i], nfp[i]– соответственно начальное (минимальное) и конечное (максимальное) числа подмножеств в разбиениях при поиске, который реализуется с использованием i-го процессора; pChi[i], np[i] – характеристический вектор разбиения, удовлетворяющий заданным ограничениям, и текущее число подмножеств в этом разбиении; icp – номер процессора, с использованием которого найдено минимальное разбиение; i, ii, – управляющие параметры циклов; k – рабочая переменная; ms[i] – сообщение i-го процессора о завершении его работы; pChiMin, npMin – характеристический вектор минимального разбиения и число подмножеств в этом разбиении.

2. Центральный процессор определяет диапазоны поиска nsp[i], nfp[i] для p процессоров, передает им эти значения и активизирует их работу:
icp =p; fp[]= (0, 0, …, 0); pChiMin =(1, 2,

…, n); k =n/p; nsp[0]=1; nfp[0]=k-1;

for(i=1;i< p;i++) { nsp[i]+= k; nfp[i]+=k; передать nsp[i], nfp[i] процессору P[i]; fp[i] =1; активизировать P[i]}

3. Активизированные процессоры осуществляют параллельный поиск минимального разбиения:

Parallel Start

for(ii=1;ii< p;ii++) { P[ii]: np[ii]= nsp[ii]; P[ii]: while (np[ii]<= nfp[ii]&& fp[ii] ==1) {Выполнить EQ2_1; if(pChi[ii] – удовлетворяет заданным ограничениям) {передать центральному процессору ms[ii]=1 и pChi ; } } P[ii]: if (fp[ii] ==1) { передать центральному процессору сообщение о завершении поиска ms[ii]=0; стоп [ii];}}

Parallel End

4. Центральный процессор, получив сообщение о завершении поиска от ii-го процессора выполняет действия:

while(fp[i]! =(0, 0, …, 0)) { Читать ms[ii]; if(ms[ii]==0) { for(i=0;i<=ii;i++) fp[i] =0;}else { Читать pChi[ii]; if(npMin > np[ii]) {icp=ii; npMin = np[ii]; pChiMin = pChi[ii]; } if (ii<n-1){for(i=ii+1;i<n;i++) fp[i] =0; }} Стоп, в pChiMin размещается характеристический вектор минимального разбиения, а в npMin – число подмножеств в минимальном разбиении.

В табл. 3 приводятся оценки временной сложности алгоритма EQ4_1 для простейшего случая: p=n. При этом считалось, что время измеряется как число рассматриваемых разбиений, и искомые варианты допустимого минимального разбиения множества X распределены равномерно.

Таблица 3

Размерность задачи	Алгоритм EQ3_1 Число рассмотренных разбиений		Алгоритм EQ4_1 Число рассмотренных разбиений		Алгоритм EQ4_1 Ускорение	
(n)	в худшем	в среднем	в худшем	в среднем	в худшем	в среднем
10	55060,1	68667,5	15585,5	42525	3,532777	1,614756
20	2,24233e+13	3,08902e+13	5,37966e+12	1,51709e+13	4,168163	2,036148
30	3,52401e+23	5,10233e+23	7,61458e+22	2,15047e+23	4,627977	2,372658
40	5,96778e+34	8,51494e+34	1,2787e+34	3,58599e+34	4,667068	2,374502
50	7,04773e+46	8,57827e+46	1,39437e+46	3,84008e+46	5,054419	2,233878

Результаты анализа, приведенные в табл. 3, показывают ускорение, которое растет по мере увеличения размерности задачи.

ВЫВОДЫ

1. Разработанный автором алгоритм EQ3_1 реализует двоичный поиск минимальных разбиений заданных множеств, что обеспечивает уменьшение временной сложности решения данной задачи в 2…3 раза.

2. Для дальнейшего уменьшения временной сложности задачи следует использовать при ее решении параллельный алгоритм EQ4_1.

ЛИТЕРАТУРА

1. *Борзунов Г.И.* // Изв. вузов. Технология текстильной промышленности. – 2008, №5. С.102…104.

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.

Рекомендована кафедрой информационных технологий и компьютерного дизайна. Поступила 23.01.09.