*Г. И. Борзунов (к. т. н., доцент)*
Московский инженерно-физический институт (государственный университет)

# СОВЕРШЕНСТВОВАНИЕ МАТЕМАТИЧЕСКОЙ МОДЕЛИ ПОИСКА ЭКСТРЕМАЛЬНЫХ РАЗБИЕНИЙ МНОЖЕСТВ

*В данной работе описываются алгоритмы, обеспечивающие ускорение поиска экстремальных разбиений в два-три раза по сравнению с известными алгоритмами. Приводятся результаты анализа временной сложности указанных алгоритмов.*

Разбиения множеств или эквивалентности, заданные на множествах, используются в качестве математических моделей при конструировании хэш-функций, применяемых в криптографии, при разработке систем обнаружения аномалий вычислительных процессов в распределенных вычислительных системах на основе TCP/IP, а также при распределении ресурсов для параллельных вычислений, которые находят все большее применение в решении задач информационной безопасности. Решение оптимизационных задач с использованием указанных математических моделей основывается на конструктивном перечислении вариантов разбиения исходного множества и проверке выполнения заданного ограничения для каждого варианта разбиения. Базовым алгоритмом такого подхода является алгоритм генерации разбиений множества. Опубликованные ранее алгоритмы [1, 2] имеют ряд преимуществ, но, как было показано в работе [3], не обеспечивают перечисление вариантов разбиений множеств в порядке монотонного возрастания числа подмножеств. В работе [3] был предложен рекурсивный алгоритм, который реализует перечисление разбиений заданного множества при монотонном возрастании числа подмножеств в разбиениях.

В данной статье рассматриваются возможности разработанного автором итеративного алгоритма Eq2_1, который также последовательно генерирует варианты разбиения множества в порядке монотонного возрастания числа подмножеств без использования рекурсии. Пусть n — число элементов в множестве X, для которого осуществляется поиск минимального разбиения, удовлетворяющего заданным ограничениям; np — число подмножеств в текущем разбиении, ic — счетчик числа рассмотренных разбиений; i, ii — управляющие параметры циклов; k — рабочая переменная; ρPsi — вектор спецификации разбиения (определение см. ниже); ρChi — характеристический вектор разбиения. Координаты вектора ρChi взаимно однозначно соответствуют элементам множества X, а числовое значение каждой из этих координат принадлежит множеству {0, 1, …, n-1} и определяет номер подмножества, которому принадлежит соответствующий элемент этого множества. Если в векторе ρChi подмножества нумеруются монотонно слева направо, то такое представление ρChi будем называть каноническим. Очевидно, что любой канонический вектор ρChi взаимно однозначно соответствует некоторому разбиению множества X. Далее будем рассматривать только канонические векторы ρChi и название «канонический» будем опускать. Пусть векторы спецификации ρPsi связаны с векторами ρChi соотношением: ρPsi[k]=max {ρChi[i] | i=0, 1, …, n-1}. Тогда любой вектор, каждая координата которого принадлежит множеству {0, 1, …, n-1}, является характеристическим вектором разбиения ρChi тогда и только тогда, когда для любого k=(0, 1, …, n-2): ρPsi[k]≥ ρPsi[k+1]+1. Кроме вышеуказанных обозначений при описании алгоритма Eq2_1 используются аргументы: ns — начальное (минимальное) значение числа подмножеств в генерируемых разбиениях, nf — конечное (максимальное) значение числа подмножеств в генерируемых разбиениях. Эти аргументы обеспечивают управление порядком генерации разбиений. Так, при ns==1 и nf==n по данному алгоритму будут построены все возможные разбиения множества, состоящего из n элементов, при ns==k, nf==k и 1<k<n будут построены только разбиения, состоящие из k подмножеств. Ниже впервые приводится псевдокод алгоритма EQ2_1, обеспечивающего конструктивное перечисление разбиений множеств при монотонном

возрастании в этих разбиениях числа подмножеств. В псевдокоде используются элементы языка программирования «Си».

1. Пусть n — число элементов в множестве X; ns — начальное (минимальное) значение числа подмножеств в разбиении; nf — конечное (максимальное) значение числа подмножеств в разбиении; nρ — текущее число подмножеств в разбиении, ic — счетчик числа рассмотренных разбиений; i, ii — управляющие параметры циклов; k — рабочая переменная; ρPsi — вектор спецификации разбиения; ρChi — характеристический вектор разбиения.

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

3. Построить ρPsi, ρChi для разбиения, состоящего из единственного подмножества: if (nρ < 2) {for(i=0;i<n;i++) ρChi[i]=ρPsi[i]=1;

nρ++; ic++; выполнить вывод ρChi}

4. Начало основного цикла. Генерировать все разбиения, в которых количество подмножеств равно nρ, nρ+1, …, nf (п. 4 — п. 11):

while(nρ<=nf) {

5. Положить значение ii равным текущему числу подмножеств в разбиениях: ii=nρ.

6. В соответствии с текущим значением nρ привести в начальное состояние ρPsi, ρChi:

for (i=n-1; i>0; i--) {if(ii>1){ρChi[i]=ρPsi[i]=ii; ii--;}else ρChi[i]=ρPsi[i]=1;}

7. Положить ic++; выполнить вывод ρChi. Положить i=n-1.

8. Выполнить построение всех остальных разбиений (ρPsi, ρChi), соответствующих текущему значению nρ (п. 9 — п. 10): while(i>0){

9. Построить очередную спецификацию (ρPsi), соответствующую текущему значению nρ, и первый из соответствующих ей характеристических векторов разбиения (ρChi):

```
for (i=n-1; i>0; i--)
  {  if((ρPsi[i]==ρPsi[i-1])&&(ρPsi[i]<nρ))
                {ρPsi[i]++; ρChi[i]=ρPsi[i]; k=nρ;
                for(ii=n-1; ii>i; ii--) {ρPsi[ii] =k; if(k>ρPsi[i]) k--;}
                for(ii=1; ii<n; ii++) {if(ρPsi[ii]==ρPsi[ii-1]) ρChi[ii]=1;
                                       else ρChi[ii]=ρPsi[ii];}
                  Положить ic++; выполнить вывод ρChi;
                  break;   } // end if(ρPsi[i]==ρPsi[i-1])&&ρPsi[i]<nρ)
     }  //end for (i=n-1; i>0; i--)
```

10. Если спецификация (ρPsi), соответствующая текущему значению nρ, построена, то построить все возможные характеристические векторы (ρChi), удовлетворяющие этой спецификации:

```
if(i>0){ii=n-1;
       while(ii>0)
            {if(ρChi[ii]<ρPsi[ii])
                  {ρChi[ii]++; k=ii; k++;
                   while(k<n){if(ρPsi[k]==ρPsi[k-1]) ρChi[k]=1; k++; }
                    ii=n-1; ic++; Выполнить вывод ρChi;
                   }//end if(ρChi[ii]<ρPsi[ii])
                    else ii--;
         }// end while(ii>0)
        } // end if(i>0) — конец п. 10.
  }  // end while(i>0) — конец п. 8.
```

11. Выполнить: nρ++; //Увеличивается число подмножеств в разбиениях.

}//end while(nρ<=n) — конец п. 3.

В таблице 1 в качестве примера приводятся результаты генерации всех возможных вариантов разбиения множества, состоящего из 6 элементов, при ns=5 и nf=5, т. е. всех возможных вариантов разбиения этого множества только на 5 подмножеств. Количество таких разбиений равно числу Стирлинга второго рода S(6,5)=15.

*Таблица 1. Генерация с помощью алгоритма Eq2_1 характеристических векторов разбиения множества, состоящего из 6 элементов, при ns=5 и nf=5.*

| № | ρPsi[i] — вектор спецификации | | | | | | ρChi[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 |

Приведенные в таблице 1 результаты показывают, что алгоритм EQ2_1 не только обеспечивает перечисление вариантов разбиений множества при монотонном возрастании числа классов разбиений, но и реализует генерацию разбиений множества на такое количество подмножеств, которое ограничивается заданным диапазоном, определяемым значениями параметров ns и nf. Это позволяет организовать поиск экстремальных разбиений в соответствии с модифицированной схемой двоичного поиска. Ниже впервые описывается алгоритм поиска минимальных разбиений заданных множеств EQ3_1, использующий модифицированную схему двоичного поиска.

Пусть исходное множество X содержит N элементов. Тогда для определения минимального разбиения множества X выполняются следующие действия:

1. Положить ns=1, nf=N, P=(0,0,0,…,0), где P — характеристический вектор допустимого разбиения множества X.

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

3. Выполнить nρ=(nf + ns)/2. Здесь деление выполняется нацело, т. е. с отбрасыванием остатка. Используя алгоритм EQ2_1, выполнить поиск допустимого разбиения множества X на nρ частей.

Если найден характеристический вектор допустимого разбиения ρChi, то выполнить: P=ρChi; иначе перейти к п. 5.

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

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

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

Приведенные ниже в таблице 2 результаты анализа алгоритмов EQ2_1, EQ3_1 показывают, что использование этих алгоритмов позволяет сократить время поиска экстремального разбиения заданного множества в два-три раза. При этом коэффициент ускорения как в среднем, так и в худшем случае возрастает одновременно с увеличением числа элементов в исходном множестве, т. е. с увеличением размерности задачи.

*Таблица 2. Сравнение коэффициентов ускорения алгоритмов EQ2_1, EQ3_1 с алгоритмами, которые не обеспечивают монотонного возрастания числа подмножеств в генерируемых разбиениях.*

| Число элементов | Ускорение алгоритма EQ2_1 по сравнению с алгоритмами [1, 2] | | Ускорение алгоритма EQ3_1 по сравнению с алгоритмом EQ2_1 | |
|---|---|---|---|---|
| | худший случай | в среднем случае | худший случай | в среднем случае |
| 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 |

Дальнейшее сокращение времени поиска экстремального разбиения заданного множества может быть достигнуто при использовании параллельного программирования. Для этого необходимо выполнить алгоритмический анализ параллельного решения указанной задачи.

## СПИСОК ЛИТЕРАТУРЫ:

1. *Романовский И. В.* Алгоритмы решения экстремальных задач. М., 1971. — 352 с.
2. *Липский В.* Комбинаторика для программистов. М., 1988. — 213 с.
3. *Борзунов Г. И., Пронин А. К.* Алгоритмы поиска экстремальных разбиений множеств большой мощности // Безопасность информационных технологий. 2006. № 1. С. 52—54.