УДК 519.85

И.В. ГРЕБЕННИК, А.С. ЛИТВИНЕНКО

ГЕНЕРАЦИЯ КОМБИНАТОРНЫХ МНОЖЕСТВ С ЗАДАННЫМИ СВОЙСТВАМИ

ВВЕДЕНИЕ

Генерация разнообразных комбинаторных объектов используется при разработке и реализации методов и алгоритмов решения многих научных и прикладных задач [1–6]. Под генерацией в них понимается построение всех комбинаторных структур определенного типа [3]. В указанных источниках главным образом решаются задачи генерации достаточно простых комбинаторных объектов — перестановок, сочетаний, разбиений, деревьев, двоичных последовательностей. Результаты генерации комбинаторных объектов используются при решении задач моделирования, комбинаторной оптимизации и в других областях [7–10]. Решение проблемы генерации более сложных комбинаторных объектов затрудняется отсутствием специальных конструктивных средств и необходимостью значительных вычислительных затрат, связанных с избыточностью результатов применения известных методов и алгоритмов генерации.

Достаточно сложные комбинаторные конфигурации могут быть формально описаны и сгенерированы с помощью конструктивных средств описания композиционных k-образов комбинаторных множеств (k-множеств), предложенных в [11, 12]. При этом под комбинаторным множеством понимается множество кортежей, построенных из конечного множества произвольных элементов (так называемых порождающих элементов) в соответствии с определенными правилами [13]. Примерами классических комбинаторных множеств могут служить перестановки, сочетания, размещения, двоичные последовательности.

Аппарат k-множеств достаточно подробно исследован в [11–13], общие принципы их генерации рассмотрены в [13]; однако сама задача генерации k-множеств в общем случае осталась нерешенной, исследован лишь один из ее частных случаев [14].

Задача генерации k-множеств, в свою очередь, требует решения задач генерации базовых комбинаторных множеств, используемых при построении k-множеств. Базовыми могут быть комбинаторные множества с известными описаниями и алгоритмами генерации: как классические комбинаторные множества (перестановки, сочетания, размещения, кортежи), так и неклассические: перестановки кортежей, композиции перестановок, перестановки с заданным количеством циклов и др. [12–14]. Для многих базовых комбинаторных множеств описаны алгоритмы их генерации [1–3, 5, 15], однако в большинстве случаев каждый алгоритм генерации основан на специфических свойствах комбинаторных множеств.

В данной работе предлагается общий подход к генерации композиционных k-образов комбинаторных множеств (k-множеств) на основе единого подхода к генерации различных базовых комбинаторных множеств.

Цель настоящей работы — решение задачи генерации композиционных k-образов комбинаторных множеств.

1. КОМПОЗИЦИОННЫЕ *k*-ОБРАЗЫ КОМБИНАТОРНЫХ МНОЖЕСТВ (*k*-МНОЖЕСТВА)

Пусть $z^{\beta} = \{z_1^{\beta}, z_2^{\beta}, z_{n_\beta}^{\beta}\} \subseteq \mathbf{Z}_{\boldsymbol{\beta}_i}$, $\mathbf{Z}_{\boldsymbol{\beta}_i}$ — множества произвольных элементов, $\beta \in \boldsymbol{\beta}_i$, $i \in J_k^0 = \{0, 1, 2, \ldots, k\}$, где $\boldsymbol{\beta}_0 = \{0\}$,

$$\boldsymbol{\beta}_i = \{\beta_j, j = 1, 2\ldots, \eta_i\},\ \beta_j = (\alpha_1, \alpha_2, \ldots, \alpha_i),$$

$$\alpha_1 \in J_n,\ \alpha_2 \in J_{n_{\alpha_1}}, \ldots, \alpha_i \in J_{n_{\alpha_1 \ldots \alpha_{i-1}}},$$

$$\eta_1 = n,\ \eta_2 = \sum_{j=1}^{n} n_j,\ \eta_i = \sum_{\alpha_1=1}^{n} \sum_{\alpha_2=1}^{n_1} \ldots \sum_{\alpha_{i-1}=1}^{n_{1\ldots i-2}} n_{\alpha_1 \ldots \alpha_{i-1}},\ i = 3, 4, \ldots, k. \quad (1)$$

Рассмотрим отображения [12, 13]

$$\Gamma_{\boldsymbol{\beta}_0} : \mathbf{Z}_{\boldsymbol{\beta}_i} \to \mathbf{Y}^0,\ \Gamma_{\boldsymbol{\beta}_i} : \mathbf{Y}^{i-1} \times \mathbf{Z}_{\boldsymbol{\beta}_i} \to \mathbf{Y}^i,$$

где $\mathbf{Y}^0 = \{Y_{\boldsymbol{\beta}_0}(z^{\beta}), \beta \in \boldsymbol{\beta}_0\}$, $\mathbf{Y}^i = \{Y_{\boldsymbol{\beta}_i}(Y_{\boldsymbol{\beta}_{i-1}}, z^{\beta}), \beta \in \boldsymbol{\beta}_i\}$, $i \in J_k$, $J_t = \{1, 2, \ldots, t\}$, $Y_{\boldsymbol{\beta}_i} = \Gamma_{\boldsymbol{\beta}_i}(Y_{\boldsymbol{\beta}_i}, \mathrm{z}^{\boldsymbol{\beta}_i}) = F(Y_{\boldsymbol{\beta}_{i-1}}, \widetilde{\Gamma}_{\boldsymbol{\beta}_i}(\mathrm{z}^{\boldsymbol{\beta}_i}))$, $i \in J_k$, $F(Y_{\boldsymbol{\beta}_{i-1}}, \widetilde{\Gamma}_{\boldsymbol{\beta}_i}(\mathrm{z}^{\boldsymbol{\beta}_i}))$ — отображение, реализующее операцию n-замещения, которая состоит в замене каждого порождающего элемента множества $Y_{\boldsymbol{\beta}_{i-1}}$ элементами базовых комбинаторных множеств $Y_{\beta} = \widetilde{\Gamma}_{\beta}(\mathbf{z}^{\beta})$, $\beta \in \boldsymbol{\beta}_i$, соответственно $\widetilde{\Gamma}_{\boldsymbol{\beta}_i}(\mathrm{z}^{\boldsymbol{\beta}_i}) = (\widetilde{\Gamma}_{\beta}(\mathrm{z}^{\beta}), \beta \in \boldsymbol{\beta}_i)$, $\mathrm{z}^{\boldsymbol{\beta}_i} = (\mathrm{z}^{\beta}, \beta \in \boldsymbol{\beta}_i)$, $\widetilde{\Gamma}_{\beta}(\mathrm{z}^{\beta})$ — базовые отображения [12, 13]. Это означает, что $(z_{l_1}^{\beta}, z_{l_2}^{\beta}, \ldots, z_{l_\beta}^{\beta}) \in Y_{\beta}$, $z_{l_t}^{\beta} \in Y_{\delta}$, $l_t \in J_{l_\beta}$, $\beta \in \boldsymbol{\beta}_{i-1}$, $\delta \in \boldsymbol{\beta}_i$, $i \in J_k$.

Обозначим

$$\boldsymbol{\Gamma}_i = \{\Gamma_{\boldsymbol{\beta}_i}\},\ i \in J_k. \quad (2)$$

Определение. Композиционным k-образом комбинаторных множеств Y_0, Y_1, $Y_2, \ldots, Y_n$, $Y_{11}, Y_{12}, \ldots, Y_{1n_1}, \ldots, Y_{\underbrace{1\ldots1}_{k}}, \ldots, Y_{nn_1\ldots n_{k-1}}$ (k-множеством), порожденным множествами z^{β_k}, $\beta_k \in \boldsymbol{\beta}_k$, называется комбинаторное множество вида [12, 13]

$$W_z = \Gamma_k \circ \Gamma_{k-1} \circ \ldots \circ \Gamma_0(z), \quad (3)$$

где отображения $\Gamma_i \in \boldsymbol{\Gamma}_i$, $i \in J_k$, определяются соотношением (2).

Мощность множества (3) определяется соотношением [12, 13]

$$Card(W_z) = \sum_{\{\gamma_1, \gamma_2, \ldots, \gamma_r\} \subset J_n} \alpha_{\gamma_1, \gamma_2, \ldots, \gamma_r} \cdot \prod_{i=1}^{k} \prod_{\beta_i = (\alpha_1 \alpha_2 \ldots \alpha_i)} Card\, Y_{\beta_i}. \quad (4)$$

Поскольку генерация k-множеств основывается на генерации базовых комбинаторных множеств, необходим алгоритм для генерации этих множеств.

2. ГЕНЕРАЦИЯ БАЗОВЫХ КОМБИНАТОРНЫХ МНОЖЕСТВ

С каждым базовым комбинаторным множеством T свяжем множество $p(T) = \{A, m, S\}$, где $A = \{a_1, a_2, \ldots, a_n\}$, $a_1 < a_2 < \ldots < a_n$, — множество порождающих элементов, m — длина кортежа $t \in T$ (полагаем, что все кортежи множества имеют равную длину), S — набор параметров, характеризующих множество T, например, параметры $n_1, n_2, \ldots, n_k$ для перестановок с повторениями и другие параметры, характерные для разных классов комбинаторных множеств. Под классом комбинаторного множества будем понимать его принадлежность перестановкам, сочетаниям и т.д.

Пусть заданы базовое комбинаторное множество T и его параметры $p(T)$. Необходимо сгенерировать все элементы $t \in T$, каждый из которых является кортежем длины m. Введем обозначение $t^i = (t_1, t_2, \ldots, t_i)$ $\forall t_i \in A$, $A \in p(T)$, $i \in J_n$. Тогда $t^0 = (\,)$ — пустой кортеж, $t^m = t \in T$. Результатом генерации является множество T.

Изложим вначале идею алгоритма генерации базовых множеств. Алгоритм работает рекурсивно, на каждом уровне рекурсии $i \in J_{m-1}^0$ добавляя в текущий кортеж $t^i = (t_1, t_2, \ldots, t_i)$ следующий элемент t_{i+1}, получая на уровне $i+1$ кортеж $t^{i+1} = (t_1, t_2, \ldots, t_{i+1})$. На уровне $m \le n$ алгоритм добавляет кортеж $t^m = t$ во множество T.

Принадлежность множества T к определенному классу комбинаторных множеств устанавливает некоторые ограничения на элемент $t_{i+1} \in A$. На каждом уровне $i \in J_{m-1}^0$ обозначим $F^i = \{f_1, f_2, \ldots, f_k\} \subseteq A$ множество всех порождающих элементов, удовлетворяющих этим ограничениям. Тогда во входной кортеж $t^i = (t_1, t_2, \ldots, t_i)$ алгоритм добавляет элемент $t_{i+1} = f_j$ для каждого $j \in J_k$ и рекурсивно вызывает себя с параметром $t^{i+1} = (t_1, t_2, \ldots, f_j)$.

Рассмотрим особенности построения множества F^i для некоторых классов комбинаторных множеств. Для размещений с повторениями во множество F^i входят все порождающие элементы: $F^i = A$.

Для размещений без повторений и перестановок (как частный случай) во множество F^i входит $n-i$ порождающих элементов, незадействованных в t^i:

$$F^i = \{f_1, f_2, \ldots, f_{n-i}\} \subseteq A : f_l \ne t_j \quad \forall j \in J_i, \forall l \in J_{n-i}.$$

Поскольку в сочетаниях порядок элементов не играет роли, будем генерировать их в виде упорядоченных наборов, для которых $t_1 < t_2 \ldots < t_i$ в случае сочетаний без повторений и $t_1 \le t_2 \le \ldots \le t_i$ для сочетаний с повторениями. Тогда для сочетаний без повторений во множество F^i входят порождающие элементы, незадействованные в t^i и большие t_i:

$$F^i = \{f_1, f_2, \ldots, f_k\} \subseteq A : f_l \ne t_j, \ f_l > t_i \quad \forall l \in J_k, \ \forall j \in J_{i-1}.$$

В случае сочетаний с повторениями в F^i также входят порождающие элементы, равные t_i:

$$F^i = \{f_1, f_2, \ldots, f_k\} \subseteq A : f_l \ne t_j, f_l \ge t_i \quad \forall l \in J_k, \forall j \in J_{i-1}.$$

Опишем алгоритм **GenBase**, реализующий описанные действия. Входные данные **GenBase** — множество T, набор параметров $p(T)$, а также кортеж t^i. На каждом уровне рекурсии $i \in J_{m-1}^0$ алгоритм формирует множество F^i, затем в t^i последовательно добавляет каждый элемент F^i, начиная с первого, и рекурсивно вызывает себя.

Для генерации всех необходимых элементов множества **GenBase** вызывается с параметрами T, $(\,)$, $p(T)$. Отметим, что базовое комбинаторное множество может состоять из единственного заданного кортежа. Тогда **GenBase** ничего не добавляет к заданному кортежу и прекращает работу:

function GenBase(T t^i, $p(T)$);

local F^i;

if $i = m$, **then** $T := T \cup t^i$; **exit**;

end if;

case T **of**

$\overline{A}_n^m : F^i = A;$

$A_n^m : F^i = \{f_1, f_2, \ldots, f_{n-i}\} \subseteq A : f_l \neq t_j, \forall j \in J_i, \forall l \in J_{n-i};$

$\overline{C}_n^m : F^i = \{f_1, f_2, \ldots, f_k\} \subseteq A : f_l \neq t_j, f_l \geq t_i, \forall l \in J_k, \forall j \in J_{i-1};$

$C_n^m : F^i = \{f_1, f_2, \ldots, f_k\} \subseteq A : f_l \neq t_j, f_l > t_i, \forall l \in J_k, \forall j \in J_{i-1};$

T_n: exit;

end case;
for $j = 1, 2, \ldots, |F^i|$ **do**

GenBase($t^{i+1} = (t_1, t_2, \ldots, t_i, f_j)$);

end for;
end function;

Для генерации комбинаторных множеств других классов с помощью данного алгоритма достаточно определить соответствующие правила формирования множества F^i.

3. ГЕНЕРАЦИЯ *k*-МНОЖЕСТВ

Пусть даны комбинаторные множества $Y_0, Y_1, Y_2 \ldots Y_n, Y_{11}, Y_{12}, \ldots, Y_{1n_1}, \ldots$ $\ldots, Y_{\underbrace{1\ldots1}_{k}}, \ldots, Y_{nn_1\ldots n_{k-1}}$, а также параметры $p(Y_0), p(Y_1), \ldots, p(Y_{nn_1\ldots n_{k-1}})$. Необходимо получить k-множество $W_z = \Gamma_k \circ \Gamma_{k-1} \circ \ldots \circ \Gamma_0(z)$, $z = A_0 \in p(Y_0)$. Введем некоторые обозначения.

Множество $Y_{i_1, i_2, \ldots, i_n}$ будем называть родительским множеством множества $Y_{j_1 j_2 \ldots j_n j_{n+1}}$, если $i_1 = j_1, i_2 = j_2, \ldots, i_n = j_n$. Множество $Y_{j_1 j_2 \ldots j_n j_{n+1}}$ будем называть дочерним множеством множества $Y_{i_1 i_2 \ldots i_n}$. Для удобства представления алгоритма решения задачи изменим систему индексации базовых множеств, перейдя от индексов переменной длины к индексам фиксированной длины. Каждое базовое множество на уровне $i \in J_{k-1}^0$ будем обозначать как Y_{ij}, где $j \in J_{\eta_i}$ — порядковый номер базового множества, а η_i определяется формулой (1). При этом для множества Y_0 в формулах будем применять обозначение Y_{01}.

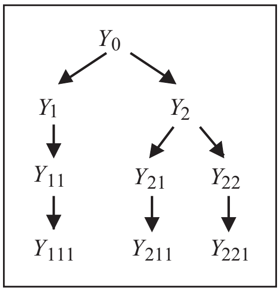

Рис. 1

Пример 1. Пусть заданы базовые множества Y_0, $Y_1, Y_2, Y_{11}, Y_{21}, Y_{22}, Y_{111}, Y_{211}, Y_{221}$, связь между которыми можно схематически представить в виде дерева (рис. 1).

На первом уровне такого дерева находятся множества Y_1 и Y_2, на втором — Y_{11}, Y_{21} и Y_{22}, на третьем — Y_{111}, Y_{211} и Y_{221}. В соответствии с новой индексацией множества первого уровня обозначим Y_{11} и Y_{12}, второго — Y_{21}, Y_{22} и Y_{23}, третьего — Y_{31}, Y_{32} и Y_{33}. Тогда схема связи базовых множеств при измененной индексации принимает вид, представленный на (рис. 2).

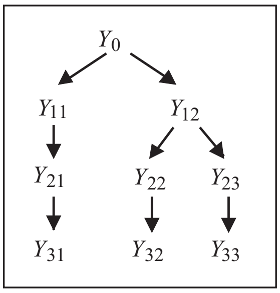

Рис. 2

Предложенный способ индексации позволяет неявно задать связь между дочерним и родительским множествами. Пусть множество Y_{ij} порождено элементами множества $A_{ij} \in p(Y_{ij})$, мощность которого $n_{ij} = |A_{ij}|$.

Тогда из способа построения k-множества следует, что на уровне $i+1$ имеется n_{ij} дочерних множеств множества Y_{ij}. Положим, что первые n_{i1} множеств на уровне $i+1$ являются дочерними множествами Y_{i1}, следующие n_{i2} множеств — дочерними для Y_{i2} и т.д. для всех Y_{ij}, $j \in J_{\eta_i}$. Тогда для каждого базового множества можно однозначно установить все его родительские и дочерние множества. При этом положим, что при выполнении операции n-замещения порождающий элемент $a_l \in A_{ij}$ родительского множества Y_{ij} будет замещен элементами l-го его дочернего множества.

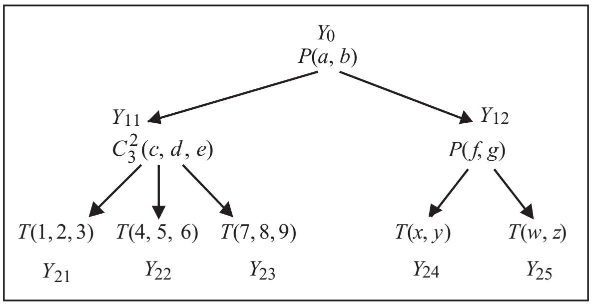

Рис. 3

Пример 2. Пусть k-множество представлено деревом (рис. 3), где $P(a, b)$ — множество перестановок элементов a и b, $C_3^2(c, d, e)$ — множество сочетаний из трех элементов по два, $T(1, 2, 3)$ — кортеж (123) и т.д.

Множество Y_{11} порождено тремя элементами: c, d, e, которые при выполнении операции n-замещения будут заменены элементами множеств Y_{21}, Y_{22} и Y_{23} соответственно. Порождающие элементы f и g множества Y_{12} будут заменены элементами множеств Y_{24} и Y_{25} соответственно.

4. АЛГОРИТМ ГЕНЕРАЦИИ *k*-МНОЖЕСТВ

Опишем алгоритм генерации k-множеств. В начале его работы с помощью алгоритма **GenBase** генерируются элементы каждого базового множества Y_{ij}, затем последовательно реализуются отображения $\Gamma_{i+1} \circ \Gamma_i \circ \ldots \circ \Gamma_0(z)$, $z = A_0 \in p(Y_0)$, для каждого $i \in J_{k-1}^0$, т.е. производится операция n-композиции, где порождающие элементы родительского множества заменяются элементами-кортежами его дочерних множеств.

На уровне $i=0$ родительским множеством является множество Y_0, дочерними — множества $Y_{11}, Y_{12}, \ldots, Y_{1\eta_1}$. На уровне $i=1$ родительское множество — результат композиции отображений $\Gamma_1 \circ \Gamma_0(z)$, полученный на нулевом уровне, дочерними множествами являются $Y_{21}, Y_{22}, \ldots, Y_{2\eta_2}$. На уровне i родительское множество — результат композиции отображений $\Gamma_i \circ \Gamma_{i-1} \circ \ldots \circ \Gamma_0(z)$, дочерние множества — $Y_{(i+1)1}, Y_{(i+1)2}, \ldots, Y_{(i+1)\eta_{i+1}}$.

Введем множество P^i, которое представляет собой результат применения композиции отображений $\Gamma_i \circ \Gamma_{i-1} \circ \ldots \circ \Gamma_0$ к исходному множеству $z = A_0 \in p(Y_0)$. Обозначим $P^i = \Gamma_i \circ \Gamma_{i-1} \circ \ldots \circ \Gamma_0(z)$, множество его порождающих элементов — A^i, длину каждого его элемента-кортежа — m^i. Так как на уровне i множество P^i состоит из элементов-кортежей множеств $Y_{i1}, \ldots, Y_{i\eta_i}$, то его порождающее множество имеет вид

$$A^i = \bigcup_{j=1}^{\eta_i} A_{ij}, \; A_{ij} \in p(Y_{ij}), \tag{5}$$

а длина каждого его элемента-кортежа равна

$$m^i = \bigcup_{j=1}^{\eta_i} m_{ij}, \; m_{ij} \in p(Y_{ij}), \tag{6}$$

где A_{ij} — множество порождающих элементов, m_{ij} — длина каждого элемента-кортежа базового множества Y_{ij}.

Для нулевого уровня $P^0 = Y_0$, $A^0 = A_0 \in p(Y_0)$, $m^0 = m_0 \in p(Y_0)$.

Для вычисления η_i при новой индексации можно использовать формулу

$$\eta_i = \sum_{j=1}^{\eta_{i-1}} | A_{(i-1)j} |, \ A_{(i-1)j} \in p(Y_{(i-1)j}), \tag{7}$$

аналогичную (1).

В операции n-замещения элементарной операцией является замена одного кортежа другим. Опишем функцию **replace**, выполняющую такую замену. Входом функции является кортеж $x = (x_1, x_2, \ldots, x_m)$, его длина m, множество элементов $A = \{a_i\}$, которыми порожден кортеж x, и множество $R = \{r_i\}$, $i \in J_n$, $n = | A |$ такое, что каждый элемент $a_i \in A$ необходимо заменить на $r_i \in R$. Выходом функции является кортеж *x*, где выполнены все замены. Функция **replace** приведена ниже:

function replace(x, m, A, R);
for $i := 1, 2, \ldots, | A |$
  for $j := 1, 2, \ldots, m$
    if $x_j = a_i$ **then** $x_j := r_i$, ;
    end if;
  end for;
end for;
return x;
end function;

Входными данными алгоритма **Gen_ k-set** генерации k-множества являются число k (число уровней дерева равно $k+1$), классы базовых множеств Y_{ij} и их параметры $p(Y_{ij})$, $i \in J_k^0$, $j \in J_{\eta_i}$. Выходом алгоритма является последовательность элементов генерируемого k-множества.

Алгоритм **Gen_ k-set** генерации *k*-множества приведен ниже:

procedure Gen_ k-set;
$\eta_0 := 1$;
for $i := 0, 1, \ldots, k$ **do**
  if $i \neq 0$ **then** $\eta_i := \sum_{j=1}^{\eta_{i-1}} | A_{(i-1)j} |$;
  for $j := 1, 2, \ldots, \eta_i$ **do**
    $Y_{ij} :=$ **Gen_ Base**$((\ ),\ Y_{ij},\ p(Y_{ij}))$;
  end for;
end for;
$P^0 := Y_0$; $A^0 := A \in p(Y_0)$; $m^0 := m \in p(Y_0)$;
for $i := 0, 1, \ldots, k-1$ **do**
  $P^{i+1} := \varnothing$;
  $A^i = \bigcup_{j=1}^{\eta_i} A_{ij}$;

$m^i = \bigcup\limits_{j=1}^{\eta_i} m_{ij}$;

foreach $x \in P^i$ **do**

  foreach $p_1 \in Y_{(i+1)1}$

    foreach $p_2 \in Y_{(i+1)2}$

     ……

      foreach $p_{\eta_{i+1}} \in Y_{(i+1)\eta_{i+1}}$

      $P^{i+1} := P^{i+1} \cup$ **replace**$(x,\ m^i,\ A^i,\ \{p_1, p_2, \ldots, p_{\eta_{i+1}}\})$;

      end for;

     ……

    end for;

  end for;

end for;

if $i+1=k$ **then print** P^{i+1};

end for;

end procedure;

На каждом уровне $i \in J_{k-1}^0$ в каждом кортеже текущего родительского множества P^i его элементы-кортежи заменяются всеми возможными комбинациями элементов-кортежей дочерних множеств $Y_{(i+1)1}, Y_{(i+1)2}, \ldots, Y_{(i+1)\eta_{i+1}}$. Проходя в цикле по всем элементам-кортежам текущего дочернего множества, каждый раз получаем набор кортежей $\{p_1, p_2, \ldots, p_{\eta_{i+1}}\}$, где $p_j \in Y_{(i+1)j}$ — текущий кортеж j-го дочернего множества, и с помощью функции **replace** в кортеже $x \in P^i$ заменяем каждый порождающий элемент $a_j \in A^i$ на $p_j \in Y_{(i+1)j}$.

Существенным преимуществом алгоритма является возможность получения промежуточных результатов, т.е. множеств P^i на каждом уровне $i \in J_{k-1}$, которые можно рассматривать как k-множества, где $k=i$.

Пример 3. Пусть необходимо сгенерировать множество перестановок двух заданных кортежей: $T_1 = (1, 2), T_2 = (3, 4)$. Тогда $k=1$ и множество Y_0 — это множество перестановок двух элементов или размещений из двух элементов по два, а $m_0 = 2$. Порождающие элементы Y_0 могут быть любыми, поэтому положим $A_0 = \{a, b\}$. Тогда $Y_0 = \{(a, b), (b, a)\}$. Множества $Y_{11} = \{(1, 2)\}$, $Y_{12} = \{(3, 4)\}$, т.е. заданные кортежи T_1 и T_2, имеют параметры $A_{11} = \{1, 2\}$, $A_{12} = \{3, 4\}$, $m_{11} = m_{12} = 2$. Представим структуру k-множества для примера 3 (рис. 4).

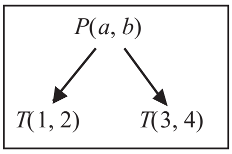

Рис. 4

В операции n-замещения в родительском множестве Y_0 первый порождающий элемент a будет заменен единственным элементом множества Y_{11}, т.е. кортежем (1, 2), а элемент b — кортежем (3, 4).

Рассмотрим процесс получения множества $W_z = \Gamma_1 \circ \Gamma_0(z) = P^1$:

1. $x = (ab)$

  1.1. $p_1 = (1, 2)$

    1.1.1. $p_2 = (34)$

    $P^1 := \varnothing \cup$ **replace**$((ab)2, \{a, b\}, \{(12), (34)\})$;

    $// P^1 = \{(1234)\}$

2. $x = (ba)$

2.1. $p_1 = (12)$

2.1.1. $p_2 = (34)$

$P^1 := \{(1234)\} \cup$ **replace**$((ba), 2, \{a, b\}, \{(12), (34)\})$;

$// P^1 = \{(1234), (3412)\}$

В результате получим $W_z = \Gamma_1 \circ \Gamma_0(z) = P^1 = \{(1234), (3412)\}$.

4. ОЦЕНКА СЛОЖНОСТИ АЛГОРИТМА Gen_ k-set

Сложность данного алгоритма пределяется сложностью генерации базовых множеств, а также сложностью выполнения операций *n*-замещения и количеством уровней k-множества.

Для генерации базовых множеств могут использоваться как известные алгоритмы с известными оценками сложности (например, [1–3, 15]), так и описанный в данной работе алгоритм **GenBase**. В любом случае каждое базовое множество Y_{ij}, $i \in J_k^0$, $j \in J_{\eta_i}$, должно быть сгенерировано одним из алгоритмов. Тогда сложность генерации базовых множеств определяется как

$$\sum_{i=0}^{k} \sum_{j=1}^{\eta_i} O(Y_{ij}), \tag{8}$$

где $O(Y_{ij})$ — сложность генерации базового множества Y_{ij}, зависящая от используемого алгоритма. Оценим $O(Y_{ij})$ для алгоритма **GenBase**.

Исходя из способов оценки сложности рекурсивного алгоритма, следуя [5], под сложностью алгоритма будем подразумевать количество изменений промежуточных данных, проведенных с момента вызова алгоритма до окончания его работы. В алгоритме **GenBase** ко входному кортежу t^i на каждом уровне рекурсии $i \in J_{m-1}^0$ добавляются элементы множества F^i и происходит рекурсивный вызов **GenBase**, т.е изменяются промежуточные данные (кортежи t^i), и поэтому сложность может быть оценена как количество рекурсивных вызовов **GenBase** на уровнях $i \in J_{m-1}^0$.

Поскольку на каждом уровне рекурсии в кортеж t^i добавляется один элемент, для генерации каждого из N кортежей $t^m = t \in T$ **GenBase** вызывается не более *m* раз. Тогда сложность генерации элементов одного базового множества не превышает mN, $N = Card(T)$.

Таким образом, если алгоритм **GenBase** используется для генерации всех базовых множеств, формула (8) принимает вид

$$\sum_{i=0}^{k} \sum_{j=1}^{\eta_i} m_{ij} Card(Y_{ij}), \; m_{ij} \in p(Y_{ij}). \tag{9}$$

Следуя выбранному способу оценки сложности, сложность выполнения операций n-замещения в алгоритме **Gen_ k-set** будем определять количеством вызовов функции **replace**, изменяющей промежуточные данные алгоритма, а именно кортежи $x \in P^i$. Из алгоритма **Gen_ k-set** следует, что на каждом уровне k-множества $i \in J_0^{k-1}$ функция **replace** вызывается

$$M = Card(P^i)\, Card(Y_{(i+1)1})\, Card(Y_{(i+1)2}) \ldots Card(Y_{(i+1)\eta_{i+1}}) \tag{10}$$

раз. Величина $Card(P^i)$ может быть получена из (4) при подстановке $k = i$.

Величины $Card\, Y_{(i+1)j}$, $j \in J_{\eta_{i+1}}$ — это мощности базовых множеств, определяемые по известным для каждого класса комбинаторных множеств формулам.

На всех уровнях $i \in J_0^{k-1}$ количество вызовов функции **replace** равно

$$\sum_{i=0}^{k-1} \left(Card(P^i) \prod_{j=1}^{\eta_{i+1}} Card(Y_{(i+1)j}) \right). \tag{11}$$

Заметим, что сложность алгоритма построения k-множества не зависит от сложности алгоритмов, используемых для генерации базовых множеств.

Таким образом, справедливо следующее утверждение.

Теорема. Вычислительная сложность алгоритма генерации k-множеств **Gen_ k-set** определяется как

$$\sum_{i=0}^{k} \sum_{j=1}^{\eta_i} O(Y_{ij}) + \sum_{i=0}^{k-1} \left(Card(P^i) \prod_{j=1}^{\eta_{i+1}} Card(Y_{(i+1)j}) \right). \tag{12}$$

Случай, когда все базовые множества являются перестановками, определен в [13] как k-композиция перестановок, получена формула для количества элементов в k-множестве. Из [13] можно получить формулу для величины $Card(P^i)$. Если все множества Y_{ij} являются перестановками n_{ij} элементов, то $Card(Y_{ij}) = n_{ij}\,!$ и

$$Card(P^i) = \prod_{u=0}^{i} \prod_{j=1}^{\eta_i} (n_{uj})!. \tag{13}$$

Тогда для k-композиции перестановок формула (10) принимает вид

$$\sum_{i=0}^{k-1} \left(\prod_{u=0}^{i} \prod_{j=1}^{\eta_i} (n_{uj})! \cdot \prod_{v=1}^{\eta_{i+1}} (n_{(i+1)v})! \right). \tag{14}$$

Следствие. Вычислительная сложность алгоритма генерации k-композиции перестановок составляет

$$\sum_{i=0}^{k} \sum_{j=1}^{\eta_i} O(Y_{ij}) + \sum_{i=0}^{k-1} \left(\prod_{u=0}^{i} \prod_{j=1}^{\eta_i} (n_{uj})! \cdot \prod_{v=1}^{\eta_{i+1}} (n_{(i+1)v})! \right), \tag{15}$$

где n_{ij} — количество порождающих элементов множества Y_{ij}.

5. ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ

На основе предложенного алгоритма решения задачи разработано программное обеспечение, с помощью которого можно генерировать k-множества различной структуры и сложности. Продемонстрируем результат работы программы для k-множества из примера 2.

В конце нулевой итерации алгоритма **Gen_ k-set** получим

$$P^1 = \{(cdfg), (cdgf), (cefg), (cegf), (defg), (degf), (fgcd), (gfcd), (fgce), (gfce), (fgde), (gfde)\}.$$

Здесь последние шесть кортежей являются перестановками пар в первых шести кортежах, что соответствует перестановкам элементов a и b во множестве Y_0.

В конце первой итерации **Gen_ k-set** получим

$$P^2 = W_z = \{(123456xywz), (123456wzxy), (123789xywz), (123789wzxy), (456789xywz), (456789wzxy), (xywz123456), (wzxy123456), (xywz123789), (wzxy123789), (xywz456789), (wzxy456789)\}.$$

Из (8) следует, что для генерации базовых множеств нужно $(2 \cdot 2) + (2 \cdot 3 + 2 \cdot 2) + (3 \cdot 1 + 3 \cdot 1 + 3 \cdot 1 + 3 \cdot 1 + 3 \cdot 1) = 29$ вызовов функции **GenBase**. По формуле (10) вызов функции **replace** производился $(2 \cdot (3 \cdot 2)) + (12 \cdot (1 \cdot 1 \cdot 1 \cdot 1 \cdot 1)) = 24$ раза. Полученное k-множество содержит 12 элементов, что совпадает с расчетами по формуле (4).

ЗАКЛЮЧЕНИЕ

В настоящей работе предложен общий подход к генерации композиционных k-образов комбинаторных множеств (k-множеств) на основе единого подхода к генерации базовых комбинаторных множеств.

Алгоритм генерации базовых множеств позволяет генерировать всевозможные комбинаторные множества, для которых могут быть заданы правила формирования множества F^i. Если такие правила задать не удается, алгоритм генерации k-множеств допускает применение других известных алгоритмов для генерации базовых множеств.

Преимуществом предложенного алгоритма для генерации k-множеств является возможность получения промежуточных результатов генерации, т.е. множеств P^i на уровнях, меньших k.

Довольно трудно определить явную зависимость времени выполнения алгоритма генерации от входных данных, так как базовые множества могут принадлежать к различным классам, имеющим различные свойства и зависимости размерности от входных данных. Однако для характерных случаев, таких как композиция перестановок, такая зависимость может быть получена. Разработанное программное обеспечение позволяет генерировать k-множества различной сложности.

СПИСОК ЛИТЕРАТУРЫ

1. Knuth D. The art of computer programming. Vol. 4. Fascicle 2: Generating all tuples and permutations. — Boston: Addison-Wesley, 2005. — 144 p.
2. Knuth D.The art of computer programming. Vol. 4. Fascicle 3: Generating all combinations and partitions. — Boston: Addison-Wesley, 2005. — 160 p.
3. Kreher D.L., Stinson D.R. Combinatorial algorithms: Generation enumeration and search. — CRC Press, 1999. — 329 p.
4. Bona M. Combinatorics of permutations. — Boston: Chapman Hall-CRC, 2004. — 383 p.
5. Ruskey F. Combinatorial generation, Dept. of Comput. Sci. Univ. of Victoria, Canada, 1j-CSC 425/20. — 2003. — 289 p.
6. Korsh J.F., LaFollette P.S. Loopless array generation of multiset permutations // The Comput. Journ. — 2004. — **47**, N 5. — P. 612–621.
7. Семенова Н.В., Колечкина Л.Н. Многокритериальные задачи комбинаторной оптимизации на множестве полиразмещений: полиэдральный подход к решению // Кибернетика и системный анализ. — 2009. — № 3. — С. 118–126.
8. Емец О.А., Емец Е.М. Модификация метода комбинаторного отсечения в задачах оптимизации на вершинно расположенных множествах // Там же. — 2009. — № 5. — С. 129–136.
9. Донец Г.А., Колечкина Л.Н. Метод упорядочения значений линейной функции на множестве перестановок // Там же. — 2009. — № 2. — С. 50–61.
10. Гребенник И.В., Панкратов А.В., Чугай А.М., Баранов А.В. Упаковка n-мерных параллелепипедов с возможностью изменения их ортогональной ориентации в n-мерном параллелепипеде // Там же. — 2010. — № 5. — С. 122–131.
11. Стоян Ю.Г., Гребенник И.В. Композиционные образы комбинаторных множеств и некоторые их свойства // Проблемы машиностроения. — 2005. — **8**, № 3. — С. 56–62.
12. Стоян Ю.Г., Гребенник И.В. Описание классов комбинаторных конфигураций на основе отображений // Доп. НАН України. — 2008. — № 10. — С. 28 –31.
13. Stoyan Yu.G., Grebennik I.V. Description and generation of combinatorial sets having special characteristics // Intern. J. of Biomed. Soft Comput. and Human Sci., Spec. Vol. «Bilevel Programming, Optimization Methods, and Applications to Economics». — 2011. — **18**, N 1. — P. 85–90.
14. Гребенник И.В. Описание и генерация перестановок, содержащих циклы // Кибернетика и системный анализ. — 2010. — № 6. — С. 97–105.
15. Липский В. Комбинаторика для программистов. — М.: Мир, 1988. — 213 с.

Поступила 29.03.2011