Cybernetics and Systems Analysis, Vol. 48, No. 6, November, 2012

GENERATING COMBINATORIAL SETS WITH GIVEN PROPERTIES

I. V. Grebennik[a†] **and O. S. Lytvynenko**[a‡]

UDC 519.85

Abstract. *Special classes of combinatorial sets called k-sets are analyzed. An algorithm for the generation of k-sets is proposed. It is based on a unified algorithm for generating base combinatorial sets. The possibilities of using it to generate various base sets are considered. The complexity of the algorithms is assessed. The results of computational experiments are analyzed.*

Keywords: *combinatorial set, generation, base set.*

INTRODUCTION

Various combinatorial objects are generated in developing and implementing solution methods and algorithms for many scientific and applied problems [1–6]. Generating them means constructing all combinatorial structures of certain type [3]. The above-mentioned publications deal mainly with problems of generating rather simple combinatorial objects such as permutations, combinations, partitions, trees, and binary sequences. The results of generation of combinatorial objects are used to solve problems of modeling, combinatorial optimization, etc. [7–10]. A lack of special constructive means and high computational cost due to redundant results of the application of well-known generation methods and algorithms make it difficult to generate more complex combinatorial objects.

Rather complex combinatorial configurations can formally be described and generated using constructive means of the description of compositional k-images of combinatorial sets (k-sets) proposed in [11, 12]. A combinatorial set is understood as a set of tuples constructed from a finite set of arbitrary elements (so-called generating elements) according to certain rules [13]. Permutations, combinations, arrangements, and binary sequences are examples of classical combinatorial sets.

The apparatus of k-sets is analyzed in detail in [11–13] and the general principles of their generation are considered in [13]; however, the problem of generating k-sets remains unsolved in the general case, only one of its special cases [14] is investigated.

In turn, the problem of generating k-sets requires problems of generating base combinatorial sets used in constructing k-sets to be solved. The base sets may be combinatorial sets with the known descriptions and generation algorithms: both classical combinatorial sets (permutations, combinations, arrangements, tuples) and nonclassical ones (permutations of tuples, compositions of permutations, permutations with a prescribed number of cycles, etc.) [12–14]. Generation algorithms are described for many base combinatorial sets [1–3, 5, 15]; however, in most cases, each generation algorithm is based on the specific properties of the combinatorial sets.

In the paper, we propose a general approach to generating compositional k-images of combinatorial sets (k-sets) based on a unified approach to generating various base combinatorial sets.

The purpose of the study is to solve the problem of generating compositional k-images of combinatorial sets.

[a]Kharkov National University of Radio Electronics, Kharkov, Ukraine, [†]*grebennik@onet.com.ua*; [‡]*litvinenko1706@gmail.com*. Translated from Kibernetika i Sistemnyi Analiz, No. 6, November–December, 2012, pp. 96–105. Original article submitted March 29, 2011.

 1060-0396/12/4806-0890

1. COMPOSITIONAL *k*-IMAGES OF COMBINATORIAL SETS (*k*-SETS)

Let $z^{\beta} = \{z_1^{\beta}, z_2^{\beta}, \ldots, z_{n_\beta}^{\beta}\} \subseteq \mathbf{Z}_{\boldsymbol{\beta}_i}, \mathbf{Z}_{\boldsymbol{\beta}_i}$ be the sets of arbitrary elements, $\beta \in \boldsymbol{\beta}_i$, $i \in J_k^0 = \{0, 1, 2, \ldots, k\}$, where $\boldsymbol{\beta}_0 = \{0\}$,

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

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

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

Let us consider the mappings [12, 13]

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

where $\mathbf{Y}^0 = \{Y_{\boldsymbol{\beta}_0}(z^{\beta}), \beta \in \boldsymbol{\beta}_0\}$, $\mathbf{Y}^i = \{Y_{\boldsymbol{\beta}_i}(Y_{\boldsymbol{\beta}_{i-1}}, z^{\beta}), \beta \in \boldsymbol{\beta}_i\}$, $i \in J_k$, $J_t = \{1, 2, \ldots, t\}$, $Y_{\boldsymbol{\beta}_i} = \Gamma_{\boldsymbol{\beta}_i}(Y_{\boldsymbol{\beta}_{i-1}}, \mathrm{z}^{\boldsymbol{\beta}_i}) = F(Y_{\boldsymbol{\beta}_{i-1}}, \widetilde{\Gamma}_{\boldsymbol{\beta}_i}(\mathrm{z}^{\boldsymbol{\beta}_i}))$, $i \in J_k$, $F(Y_{\boldsymbol{\beta}_{i-1}}, \widetilde{\Gamma}_{\boldsymbol{\beta}_i}(\mathrm{z}^{\boldsymbol{\beta}_i}))$ is a mapping implementing an n-replacement operation, which replaces each generating element of the set with $Y_{\boldsymbol{\beta}_{i-1}}$ elements of the base combinatorial sets $Y_{\beta} = \widetilde{\Gamma}_{\beta}(\mathrm{z}^{\beta})$, $\beta \in \boldsymbol{\beta}_i$, respectively, $\widetilde{\Gamma}_{\boldsymbol{\beta}_i}(\mathrm{z}^{\boldsymbol{\beta}_i}) = (\widetilde{\Gamma}_{\beta}(\mathrm{z}^{\beta}), \beta \in \boldsymbol{\beta}_i)$, $\mathrm{z}^{\boldsymbol{\beta}_i} = (\mathrm{z}^{\beta}, \beta \in \boldsymbol{\beta}_i)$, and $\widetilde{\Gamma}_{\beta}(\mathrm{z}^{\beta})$ are base mappings [12, 13]. This means that $(z_{l_1}^{\beta}, z_{l_2}^{\beta}, \ldots, z_{l_\beta}^{\beta}) \in Y_{\beta}$, $z_{l_t}^{\beta} \in Y_{\delta}$, $l_t \in J_{l_\beta}$, $\beta \in \boldsymbol{\beta}_{i-1}$, $\delta \in \boldsymbol{\beta}_i$, $i \in J_k$.

Denote

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

Definition. The compositional k-image of combinatorial sets 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-set) generated by sets z^{β_k}, $\beta_k \in \boldsymbol{\beta}_k$, is a combinatorial set [12, 13]

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

where the mappings $\Gamma_i \in \boldsymbol{\Gamma}_i$, $i \in J_k$, are defined by (2).

The cardinality of set (3) is defined by [12, 13]

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

Since generation of k-sets is based on generation of base combinatorial sets, an algorithm to generate the latter is necessary.

2. GENERATING BASE COMBINATORIAL SETS

Let us associate each base combinatorial set T with a set $p(T) = \{A, m, S\}$, where $A = \{a_1, a_2, \ldots, a_n\}$, $a_1 < a_2 < \ldots < a_n$, is the set of generating elements, m is the length of a tuple $t \in T$ (all the tuples of the set are assumed to be of the same length), S is a set of the parameters that characterize the set T, for example, parameters $n_1, n_2, \ldots, n_k$ for permutations with repetitions and other parameters typical for different classes of combinatorial sets. By the class of a combinatorial set we will mean its membership in permutations, combinations, etc.

Suppose we are given a base combinatorial set T and its parameters $p(T)$. It is necessary to generate all the elements $t \in T$, each being a tuple of length m. Let us introduce the notation $t^i = (t_1, t_2, \ldots, t_i)$ $\forall t_i \in A$, $A \in p(T)$, $i \in J_n$. Then $t^0 = (\,)$ is an empty tuple, $t^m = t \in T$. The result of the generation is the set T.

Let us first present the idea of the algorithm of generating base sets. The algorithm is recursive. At each level of recursion $i \in J_{m-1}^0$, it supplements a current tuple $t^i = (t_1, t_2, \ldots, t_i)$ with the next element t_{i+1} and obtains tuple $t^{i+1} = (t_1, t_2, \ldots, t_{i+1})$ at the level $i+1$. At the level $m \le n$, the algorithm adds tuple $t^m = t$ to the set T.

The membership of the set T to a certain class of combinatorial sets imposes some constrains on the element $t_{i+1} \in A$. At each level $i \in J_{m-1}^0$, denote by $F^i = \{f_1, f_2, \ldots, f_k\} \subseteq A$ the set of all generating elements that satisfy these constraints. Then in the input tuple $t^i = (t_1, t_2, \ldots, t_i)$ the algorithm adds element $t_{i+1} = f_j$ for each $j \in J_k$ and recursively calls itself with parameter $t^{i+1} = (t_1, t_2, \ldots, f_j)$.

Let us consider the features of constructing the set F^i for some classes of combinatorial sets. For arrangements with repetitions, the set F^i contains all the generating elements: $F^i = A$.

For arrangements without repetitions and permutations (as a special case), the set F^i contains $n-i$ generating elements not involved in t^i:

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

Since the order of elements is of no importance in combinations, we will generate them as ordered sets for which $t_1 < t_2 < \ldots < t_i$ in the case of combinations without repetitions and $t_1 \le t_2 \le \ldots \le t_i$ for combinations with repetitions. Then for combinations without repetitions the set F^i contains generating elements that do not appear in t^i and large t_i:

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

In case of combinations with repetitions, F^i also contains generating elements equal to t_i:

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

Let us describe the **GenBase** algorithm that implements these operations. The input data for **GenBase** are set T, set of parameters $p(T)$, and tuple t^i. At each level of recursion $i \in J_{m-1}^0$, the algorithm forms set F^i, then sequentially adds each element of F^i, beginning with the first one, into t^i and recursively calls itself.

To generate all the necessary elements of the set, **GenBase** is called with parameters T, ( ), and $p(T)$. Note that the base combinatorial set may consist of a unique given tuple. Then **GenBase** adds nothing to the given tuple and stops:

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**
$\overline{A}_n^m : F^i = A$;
$A_n^m : F^i = \{f_1, f_2, \ldots, f_{n-i}\} \subseteq A : f_l \ne t_j, \forall j \in J_i, \forall l \in J_{n-i}$;
$\overline{C}_n^m : F^i = \{f_1, f_2, \ldots, f_k\} \subseteq A : f_l \ne t_j, f_l \ge t_i, \forall l \in J_k, \forall j \in J_{i-1}$;
$C_n^m : F^i = \{f_1, f_2, \ldots, f_k\} \subseteq A : f_l \ne t_j, f_l > t_i, \forall l \in J_k, \forall j \in J_{i-1}$;
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;

To generate combinatorial sets of other classes by this algorithm, it will suffice to define the corresponding rules of forming the set F^i.

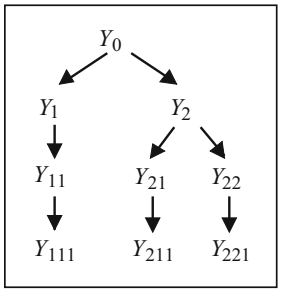

Fig. 1

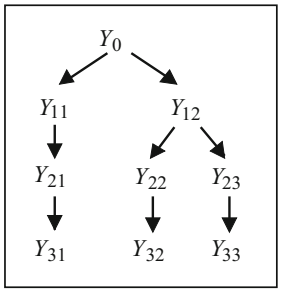

Fig. 2

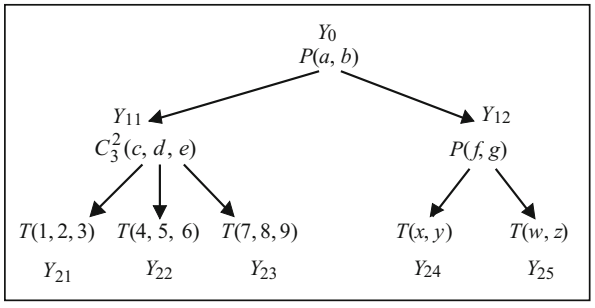

Fig. 3

3. GENERATING *k*-SETS

Suppose we are given combinatorial sets $Y_0, Y_1, Y_2 \ldots Y_n, Y_{11}, Y_{12}, \ldots, Y_{1n_1}, \ldots, Y_{\underbrace{1\ldots1}_{k}}, \ldots, Y_{nn_1\ldots n_{k-1}}$ and parameters $p(Y_0), p(Y_1), \ldots, p(Y_{nn_1\ldots n_{k-1}})$. It is necessary to obtain a *k*-set $W_z = \Gamma_k \circ \Gamma_{k-1} \circ \ldots \circ \Gamma_0(z)$, $z = A_0 \in p(Y_0)$. Let us introduce a notation.

A set $Y_{i_1 i_2 \ldots i_n}$ is called parent set of the set $Y_{j_1 j_2 \ldots j_n j_{n+1}}$ if $i_1 = j_1, i_2 = j_2, \ldots, i_n = j_n$. A set $Y_{j_1 j_2 \ldots j_n j_{n+1}}$ is called a daughter set of the set $Y_{i_1 i_2 \ldots i_n}$. For convenience sake, we will change the indexing of the base sets by passing from indices of variable length to indices of fixed length. We will denote each base set at a level $i \in J_{k-1}^0$ by Y_{ij}, where $j \in J_{\eta_i}$ is the serial number of the base set and η_i is defined (1). We will denote the set Y_0 by Y_{01} in the formulas.

Example 1. Let the base sets $Y_0, Y_1, Y_2, Y_{11}, Y_{21}, Y_{22}, Y_{111}, Y_{211}$, and Y_{221} be given, and the relations among them can be schematized as a tree (Fig. 1).

The sets Y_1 and Y_2 are at the first level of such a tree, Y_{11}, Y_{21}, and Y_{22} are at the second level, and Y_{111}, Y_{211}, and Y_{221} are at the third level. According to the new indexing, we denote the sets of the first level by Y_{11} and Y_{12}, of the second by Y_{21}, Y_{22}, and Y_{23}, and of the third by Y_{31}, Y_{32}, and Y_{33}. With the modified indexing, the base sets are related as in Fig. 2.

The proposed indexation allows specifying implicitly the relations between daughter and parent sets. Let the set Y_{ij} be generated by elements of the set $A_{ij} \in p(Y_{ij})$ whose cardinality is $n_{ij} = |A_{ij}|$. Then it follows from the construction of the *k*-set that there are n_{ij} daughter sets of the set Y_{ij} at the level $i+1$. Suppose that the first n_{i1} sets at the level $i+1$ are daughter sets of Y_{i1}, the next n_{i2} sets are daughter sets for Y_{i2}, etc., for all Y_{ij}, $j \in J_{\eta_i}$. Then for each base set, all its parent and daughter sets can be established uniquely. Suppose that during the operation of *n*-replacement, generating element $a_l \in A_{ij}$ of the parent set Y_{ij} will be replaced with elements of its *l*th daughter set.

Example 2. Let a *k*-set be presented as a tree (Fig. 3), where $P(a, b)$ is the set of permutations of elements *a* and *b*, $C_3^2(c, d, e)$ is the set of combinations of two elements out of three, $T(1, 2, 3)$ is a tuple (123), etc.

Set Y_{11} is generated by three elements: c, d, and *e*. During the operation of *n*-replacement, they will be replaced with elements of the sets Y_{21}, Y_{22}, and Y_{23}, respectively. The generating elements f and g of the set Y_{12} will be replaced with elements of the sets Y_{24} and Y_{25}, respectively.

4. ALGORITHM OF GENERATING *k*-SETS

Let us describe the algorithm of generating *k*-sets. In the beginning, the **GenBase** algorithm is used to generate elements of each base set Y_{ij}, then mappings $\Gamma_{i+1} \circ \Gamma_i \circ \ldots \circ \Gamma_0(z)$, $z = A_0 \in p(Y_0)$ are implemented sequentially for each $i \in J^0_{k-1}$, i.e., the operation of *n*-composition is carried out, where generating elements of the parent set are replaced with tuple elements of its daughter sets.

At the level $i = 0$, the parent set is the set Y_0, daughter sets are $Y_{11}, Y_{12}, \ldots, Y_{1\eta_1}$. At the level $i = 1$ the parent set is the result of the composition of mappings $\Gamma_1 \circ \Gamma_0(z)$ obtained at zero level, daughter sets are $Y_{21}, Y_{22}, \ldots, Y_{2\eta_2}$. At the level *i* the parent set is the result of the composition of mappings $\Gamma_i \circ \Gamma_{i-1} \circ \ldots \circ \Gamma_0(z)$, daughter sets are $Y_{(i+1)1}, Y_{(i+1)2}, \ldots, Y_{(i+1)\eta_{i+1}}$.

Let us introduce a set P^i, which is the result of the application of the composition of mappings $\Gamma_i \circ \Gamma_{i-1} \circ \ldots \circ \Gamma_0$ to the original set $z = A_0 \in p(Y_0)$. Denote $P^i = \Gamma_i \circ \Gamma_{i-1} \circ \ldots \circ \Gamma_0(z)$, the set of its generating elements is A^i, the length of each its tuple element is m^i. Since the set P^i at the level *i* consists of tuple elements of the sets $Y_{i1}, \ldots, Y_{i\eta_i}$, its generating set has the form

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

and the length of each its tuple element is

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

where A_{ij} is the set of generating elements, m_{ij} is the length of each tuple element of the base set Y_{ij}.

For zero level $P^0 = Y_0$, $A^0 = A_0 \in p(Y_0)$, $m^0 = m_0 \in p(Y_0)$.

To calculate η_i for the new indexing, we can use the formula

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

similar to (1).

Replacing one tuple with another is an elementary operation in the operation of *n*-replacement. Let us describe the function **replace**, which carries out such a replacement. The input of the function is a tuple $x = (x_1, x_2, \ldots, x_m)$, its length is *m*, the set of elements $A = \{a_i\}$ by which the tuple *x* is generated, and the set $R = \{r_i\}$, $i \in J_n$, $n = |A|$, such that it is necessary to replace each element $a_i \in A$ with $r_i \in R$. The output of the function is the tuple *x*, where all the replacements are made. The function **replace** is presented below:

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;

The input data of the algorithm **Gen_ k-set** of generating a *k*-set are the number *k* (the number of levels of the tree is $k+1$), classes of base sets Y_{ij}, and their parameters $p(Y_{ij})$, $i \in J^0_k$, $j \in J_{\eta_i}$. The output of the algorithm is the sequence of elements of the generated *k*-set.

Below is the algorithm **Gen_ k-set** for generating a *k*-set:

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**

$P^{i+1} := \varnothing$;

$A^i = \bigcup_{j=1}^{\eta_i} A_{ij}$;

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

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;

At each level $i \in J_{k-1}^0$ in each tuple of the current parent set P^i, its tuple elements are replaced with all possible combinations of tuple elements of the daughter sets $Y_{(i+1)1}, Y_{(i+1)2}, \ldots, Y_{(i+1)\eta_{i+1}}$. Traversing all the tuple elements of the current daughter set in the cycle, we obtain a set of tuples $\{p_1, p_2, \ldots, p_{\eta_{i+1}}\}$ each time, where $p_j \in Y_{(i+1)j}$ is the current tuple of the *j*th daughter set, and use the replace function in the tuple $x \in P^i$ to replace each generating element $a_j \in A^i$ with $p_j \in Y_{(i+1)j}$.

A significant advantage of the algorithm is that it is possible to obtain intermediate results, i.e., sets P^i at each level $i \in J_{k-1}$, which can be considered as k sets, where $k = i$.

Example 3. Let it be necessary to generate the set of permutations of two given tuples, $T_1 = (1, 2)$ and $T_2 = (3, 4)$. Then $k = 1$ and Y_0 is the set of permutations of two elements or arrangements of two elements out of two, and $m_0 = 2$. The generating elements Y_0 may be arbitrary; therefore, suppose $A_0 = \{a, b\}$. Then $Y_0 = \{(a, b), (b, a)\}$. The sets $Y_{11} = \{(1, 2)\}$, $Y_{12} = \{(3, 4)\}$, i.e., the given tuples T_1 and T_2, have the parameters $A_{11} = \{1, 2\}$, $A_{12} = \{3, 4\}$, and $m_{11} = m_{12} = 2$. Let us present the structure of the k-set for Example 3 (Fig. 4).

In the operation of *n*-replacement in the parent set Y_0, the first generating element *a* will be replaced with the unique element of the set Y_{11}, i.e., with the tuple (1,2), and element *b* with the tuple (3,4).

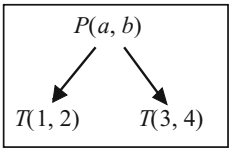

Fig. 4

Let us consider the process of deriving the set $W_z = \Gamma_1 \circ \Gamma_0(z) = P^1$:

1. $x = (ab)$

1.1. $p_1 = (1, 2)$

1.1.1. $p_2 = (34)$

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

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

2. $x = (ba)$

2.1. $p_1 = (12)$

2.1.1. $p_2 = (34)$

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

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

As a result, we obtain $W_z = \Gamma_1 \circ \Gamma_0(z) = P^1 = \{(1234), (3412)\}$.

4. ESTIMATING THE COMPLEXITY OF THE ALGORITHM Gen_ k-set

The complexity of this algorithm depends on the complexity of the generation of the base sets and on the complexity of the operations of *n*-replacement and the number of levels of the *k*-set.

To generate the base sets, both well-known algorithms with the known estimates of complexity (for example, [1–3, 15]) and the **GenBase** algorithm described in the present paper can be used. In any case, each base set Y_{ij}, $i \in J_k^0$, $j \in J_{\eta_i}$, should be generated by one of the algorithms. Then the complexity of the generation of the base sets is defined as

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

where $O(Y_{ij})$ is the complexity of the generation of the base set Y_{ij} dependent on the algorithm being used. Let us estimate $O(Y_{ij})$ for the **GenBase** algorithm.

Based on the estimates of the complexity of the recursive algorithm, following [5], by the complexity of an algorithm we will mean the number of variations in the intermediate data from the call of the algorithm till its termination. In the **GenBase** algorithm, the input tuple t^i at each level of recursion $i \in J_{m-1}^0$ is supplemented with elements of the set F^i and recursive call of the **GenBase** follows, i.e., the intermediate data (tuples t^i) vary, thus the complexity can be estimated as the number of recursive calls of **GenBase** at the levels $i \in J_{m-1}^0$.

Since one element is added to the tuple t^i at each level of recursion, to generate each of the N tuples $t^m = t \in T$ **GenBase** is called no more than m times. Then the complexity of generating elements of one base set does not exceed mN, $N = Card(T)$.

Thus, if the **GenBase** algorithm is used to generate all the base sets, formula (8) becomes

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

Following the chosen way of estimating the complexity, we will determine the complexity of operations of n-replacement in the algorithm **Gen_ k-set** depending on the number of calls of the **replace** function, which changes the intermediate data of the algorithm, namely, tuples $x \in P^i$. As follows from the **Gen_ k-set** algorithm, at each level $i \in J_0^{k-1}$ of the k-set the **replace** function is called

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

times. The value of $Card(P^i)$ can be obtained from (4) in the substitution of $k = i$. The quantities $Card\, Y_{(i+1)j}$, $j \in J_{\eta_{i+1}}$, are the cardinalities of the base sets defined by the formulas known for each class of combinatorial sets.

At all the levels $i \in J_0^{k-1}$ the **replace** function is called

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

times.

Note that the complexity of the algorithm of constructing a k-set does not depend on the complexity of the algorithms used to generate the base sets.

Thus, the following statement is true.

THEOREM. The computational complexity of the algorithm **Gen_ k-set** for generating k-sets is defined as

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

The case where all the base sets are permutations is defined in [13] as a k-composition of permutations, a formula is obtained for the number of elements in the k-set. From [13] it is possible to derive a formula for $Card(P^i)$. If all the sets Y_{ij} are permutations of n_{ij} elements, then $Card(Y_{ij}) = n_{ij}!$ and

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

Then for the k-composition of permutations formula (10) becomes

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

COROLLARY. The computational complexity of the algorithm of generating the k-composition of permutations is

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

where n_{ij} is the number of generating elements of the set Y_{ij}.

5. COMPUTATIONAL EXPERIMENTS

The proposed solution algorithm was used to develop a software to generate k-sets of different structure and complexity. Let us show the result of the program for the k-set from Example 2.

At the end of the zero iteration of the algorithm **Gen_ k-set** we obtain

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

The last six tuples are permutations of pairs in the first six tuples, which corresponds to permutations of elements a and b in the set Y_0.

At the end of the first iteration in the **Gen_ k-set** we obtain

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

As follows from (8), $(2\cdot 2)+(2\cdot 3+2\cdot 2)+(3\cdot 1+3\cdot 1+3\cdot 1+3\cdot 1+3\cdot 1)=29$ calls of the **GenBase** function are necessary to generate the base sets. By formula (10), the **replace** function was called $(2\cdot(3\cdot 2))+(12\cdot(1\cdot 1\cdot 1\cdot 1\cdot 1))=24$ times. The obtained k-set contains 12 elements, which coincides with the calculations by formula (4).

CONCLUSIONS

We have proposed a general approach to generating compositional k-images of combinatorial sets (k-sets) based on a unified approach to generating base combinatorial sets.

The algorithm to generate the base sets allows generating all possible combinatorial sets for which the rules of formation of the set F^i can be specified. If it is impossible to specify such rules, the algorithm of generating k-sets admits the application of other well-known algorithms to generate the base sets.

An advantage of the proposed algorithm of generating k-sets is that intermediate results of the generation can be obtained, i.e., sets P^i at levels lower than k.

It is rather difficult to determine an explicit dependence of the time of the generation algorithm on the input data since the base sets can belong to different classes with different properties and dependence of dimensions on the input data. However, such a dependence can be obtained for typical cases such as the composition of permutations. The software developed allows generating k-sets of different complexity.

REFERENCES

1. D. Knuth, The Art of Computer Programming, Vol. 4, Fascicle 2: Generating All Tuples and Permutations, Addison-Wesley, Boston (2005).
2. D. Knuth, The Art of Computer Programming, Vol. 4, Fascicle 3: Generating All Combinations and Partitions, Addison-Wesley, Boston (2005).
3. D. L. Kreher and D. R. Stinson, Combinatorial Algorithms: Generation Enumeration and Search, CRC Press (1999).
4. M. Bona, Combinatorics of Permutations, Chapman Hall-CRC, Boston (2004).
5. F. Ruskey, Combinatorial Generation, Dept. of Comput. Sci. Univ. of Victoria, Canada, 1j-CSC 425/20 (2003).
6. J. F. Korsh and P. S. LaFollette, "Loopless array generation of multiset permutations," The Comput. J., **47**, No. 5, 612–621 (2004).
7. N. V. Semenova and L. N. Kolechkina, " A polyhedral approach to solving multicriterion combinatorial optimization problems over sets of polyarrangements:," Cybern. Syst. Analysis, **45**, No. 3, 438–445 (2009).
8. O. A. Yemets and Ye. M. Yemets, "A modification of the method of combinatorial truncation in optimization problems over vertex-located sets," Cybern. Syst. Analysis, **45**, No. 5, 785–791 (2009).
9. G. A. Donets and L. N. Kolechkina, "Method of ordering the values of a linear function on a set of permutations," Cybern. Syst. Analysis, **45**, No. 2, 204–213 (2009).
10. I. V. Grebennik, A. V. Pankratov, A. M. Chugay, and A. V. Baranov, "Packing n-dimensional parallelepipeds with the feasibility of changing their orthogonal orientation in an n-dimensional parallelepiped," Cybern. Syst. Analysis, **46**, No. 5, 793–802 (2010).
11. Yu. G. Stoyan and I. V. Grebennik, "Compositional images of combinatorial sets and some of their properties," Problemy Mashinostr., **8**, No. 3, 56–62 (2005).
12. Yu. G. Stoyan and I. V. Grebennik, "Describing classes of combinatorial configurations based on mappings," Dop. NANU, No. 10, 28–31 (2008).
13. Yu. G. Stoyan and I. V. Grebennik, "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," **18**, No. 1, 85–90 (2011).
14. I. V. Grebennik, "Description and generation of permutations containing cycles," Cybern. Syst. Analysis, **46**, No. 6, 945–952 (2010).
15. W. Lipski, Combinatorics for Programmers [in Polish], Polish Sci. Publ. (PWN), Warsaw (1982).