ECONTECHMOD. AN INTERNATIONAL QUARTERLY JOURNAL – 2016, Vol.5, No.4, 43-48

RANDOM GENERATION OF COMBINATORIAL SETS WITH SPECIAL PROPERTIES

I. Grebennik, O. Lytvynenko

Kharkiv National University of Radio Electronics; email: igorgrebennik@gmail.com

Received October 14.2016: accepted November 15.2016

Abstract. General approach for solving the problem of random generation of compositional $k-$images of combinatorial sets (*k*-sets) has been proposed. *K*-sets are powerful apparatus that can be applied for solving many scientific and applied problems. Though many literature is dedicated to the problem of generating combinatorial configurations, existing studies deals mostly with simple combinatorial configurations like combinations, permutations etc.

The algorithms of generation both basic combinatorial sets and *k*-sets have been described. Algorithm for random generation of basic sets allows generating various combinatorial sets, and laws of constructing basic combinatorial sets can be pre-set. If identification of the laws fails, the algorithm allows using other algorithms to generate basic sets.

Complexity of described algorithms has been evaluated. The complexity of the algorithm of generation *k*-sets is determined by the complexity of generation of basic sets, as well as the complexity of operations of *n*-substitution and a number of levels of a certain *k*-set.

The described approach to the random generation is very flexible since it allows obtaining various results by varying algorithm parameters. In its turn, it allows adjusting the number of elements for both basic sets and *k*-sets. The developed software allows solving the described problems of random generation of *k* -sets and basic combinatorial sets.

Key words: combinatorial generation, k-set, basic combinatorial set, random generation, complexity.

INTRODUCTION

Generation of various combinatorial objects is frequently required in developing and implementing methods and algorithms for solving many scientific and applied problems [1-6]. Generation is usually understood as a construction of all combinatorial structures of a given type [3]. In these sources, the problem of generation of simple combinatorial objects like permutations, combinations, splits, trees, binary sequences is mainly solved. Solution of generating more complex combinatorial objects is hindered by lack of special structural tools and by significant computational costs caused by redundant results of using well-known methods and algorithms of generation.

THE ANALYSIS OF RECENT RESEARCHES AND PUBLICATIONS

Quite complex combinatorial configurations can be formally described and generated with the help of structural tools of description of compositional $k-$images of combinatorial sets ($k-$sets), proposed in [7]. A combinatorial set is understood as a variety of tuples constructed from a finite set of arbitrary elements (generating elements) in accordance with certain rules [7, 8].

Permutations, combinations, arrangements, binary sequences etc. can serve as examples of classical combinatorial sets. The apparatus of $k-$sets has been widely explored [7–9]: general concepts of their generation are considered in [8], the task of exhaustive generation of $k-$sets was solved in [9], and some special cases of this task were studied in [10].

However, the task of random generation of $k-$sets hasn't been solved yet.

Both random and exhaustive generation of $k-$sets require solving the problem of generation of basic combinatorial sets used to construct $k-$sets. Basic sets can be combinatorial ones with known descriptions and algorithms of generation: either classical combinatorial sets (permutations, combinations etc.) or non-classical, e.g., permutations of tuples, compositions of permutations, permutations with given number of cycles etc. [7-12].

Algorithms of generation of many basic combinatorial sets have been described in many articles [1-3, 5, 13-21]. However, in most cases, each generation algorithm is based on specific properties of combinatorial sets.

OBJECTIVES

Objectives of this paper are:

1. Developing a general approach for solving the problem of random generation of *k*-sets.
2. Developing the algorithms of generation of *k*-sets have been described.
3. Evaluating the complexity of constructed algorithms.

MAIN RESULTS OF THE RESEARCH

Let us briefly remind mathematical description of compositional $k-$images of combinatorial sets ($k-$sets).

COMPOSITION *K*-IMAGES OF COMBINATORIAL SETS (K-SETS)

Let $z^{\beta}=\{z_1^{\beta}, z_2^{\beta},..., z_{n_\beta}^{\beta}\}\subseteq \mathbf{Z}_{\boldsymbol{\beta}_i}$, where $\mathbf{Z}_{\boldsymbol{\beta}_i}$ are sets of arbitrary elements,

$$\beta\in\boldsymbol{\beta}_i,\ i\in J_k^0=\{0,1,...,k\},$$

where:

$$\begin{gathered}\boldsymbol{\beta}_0=\{0\},\ \boldsymbol{\beta}_i=\{\beta_j, j=1,2...,\eta_i\},\\ \beta_j=(\alpha_1,\alpha_2,...,\alpha_i),\ \alpha_1\in J_n,\\ \alpha_2\in J_{n_{\alpha_1}},...,\alpha_i\in J_{n_{\alpha_1...\alpha_{i-1}}},\\ \eta_1=n,\ \eta_2=\sum_{j=1}^{n} n_j,\\ \eta_i=\sum_{\alpha_1=1}^{n}\sum_{\alpha_2=1}^{n_1}...\sum_{\alpha_{i-1}=1}^{n_{1...i-2}} n_{\alpha_1...\alpha_{i-1}},\ i=3,4,...,k.\end{gathered}\tag{1}$$

Let us consider mappings [7, 8]:

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

where:

$$\begin{gathered}\mathbf{Y}^0=\{Y_{\boldsymbol{\beta}_0}(z^{\beta}),\beta\in\boldsymbol{\beta}_0\},\\ \mathbf{Y}^i=\{Y_{\boldsymbol{\beta}_i}(Y_{\boldsymbol{\beta}_{i-1}},z^{\beta}),\beta\in\boldsymbol{\beta}_i\},\ i\in J_k,\\ J_t=\{1,2,...,t\},\end{gathered}$$

$$Y_{\boldsymbol{\beta}_i}=\Gamma_{\boldsymbol{\beta}_i}(Y_{\boldsymbol{\beta}_{i-1}},z^{\boldsymbol{\beta}_i})=F(Y_{\boldsymbol{\beta}_{i-1}},\tilde{\Gamma}_{\boldsymbol{\beta}_i}(z^{\boldsymbol{\beta}_i})),\ i\in J_k,$$

$F(Y_{\boldsymbol{\beta}_{i-1}},\tilde{\Gamma}_{\boldsymbol{\beta}_i}(z^{\boldsymbol{\beta}_i}))$ – a mapping that realizes n-composition which consists in replacing each generative element of set $Y_{\boldsymbol{\beta}_{i-1}}$ with elements of primary combinatorial sets $Y_\beta=\tilde{\Gamma}_\beta(z^\beta),\ \beta\in\boldsymbol{\beta}_i$, respectively,

$$\tilde{\Gamma}_{\boldsymbol{\beta}_i}(z^{\boldsymbol{\beta}_i})=(\tilde{\Gamma}_\beta(z^\beta),\ \beta\in\boldsymbol{\beta}_i),\ z^{\boldsymbol{\beta}_i}=(z^\beta,\ \beta\in\boldsymbol{\beta}_i),$$

$\tilde{\Gamma}_\beta(z^\beta)$ are the primary mappings [7, 8]. It means that:

$$\begin{gathered}(z_{l_1}^{\beta},z_{l_2}^{\beta},...,z_{l_\beta}^{\beta})\in Y_\beta,\ z_{l_t}^{\beta}\in Y_\delta,\ l_t\in J_{l_\beta},\\ \beta\in\boldsymbol{\beta}_{i-1},\ \delta\in\boldsymbol{\beta}_i,\ i\in J_k.\end{gathered}$$

Let us denote

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

Definition. Composition *k*-image of combinatorial sets

$$\begin{gathered}Y_0,\ Y_1,Y_2,...,Y_n,\\ Y_{11},Y_{12},...,Y_{1n_1},\ ...,\ \underset{k}{Y_{1...1}},...,Y_{nn_1...n_{k-1}}\end{gathered}$$

($k-$set) generated by sets z^{β_k}, $\beta_k\in\boldsymbol{\beta}_k$ is the combinatorial set [7,8]:

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

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

Cardinality of the set (3) can be obtained [7, 8] as:

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

As the generation of $k-$sets is based on the generation of basic combinatorial sets, we need an algorithm for generating these sets.

GENERATION OF BASIC COMBINATORIAL SETS

Let us associate the set

$$p(T)=\{A,m,S\}$$

with each basic combinatorial set T, where

$$A=\{a_1,a_2,...,a_n\},\ a_1<a_2<...<a_n$$

is a set of generating elements, m is the length of tuple $t\in T$ (let us consider that all tuples in the set are of the same length), S is a set of parameters describing the set T (e.g., parameters $n_1,n_2,...,n_k$ for permutations with repetitions and other parameters specific for different classes of combinatorial sets). We understand a class of combinatorial sets as its belonging to permutations, combinations, etc.

Let the basic set T and its parameters $p(T)$ be defined. We need to generate all elements $t \in T$, where each element is a tuple of the length m. Let us denote

$$t^i = (t_1, t_2, ..., t_i),\ \forall t_i \in A,\ A \in p(T),\ i \in J_n.$$

It means that $t^0 = (\ )$ is an empty tuple and $t^m = t \in T$.

Firstly, let us recall the main concept of the algorithm of random generation of basic sets [9]. This algorithm is of a recursive nature: at each recursion level $i \in J^0_{m-1}$ it expands current tuple $t^i = (t_1, t_2, ..., t_i)$ by adding the following element t_{i+1} and thus obtaining a tuple $t^{i+1} = (t_1, t_2, ..., t_{i+1})$ at level $i+1$. At level $m \le n$ the algorithm adds tuple $t^m = t$ to the resulting set *T*.

The fact that set *T* belongs to a certain class of combinatorial sets imposes some restrictions on element $t_{i+1} \in A$.

At each level $i \in J^0_{m-1}$ let

$$F^i = \{f_1, f_2, ..., f_k\} \subseteq A$$

denote a set of all generating elements that satisfy these restrictions. In this case, for each $j \in J_k$, the algorithm adds element $t_{i+1} = f_j$ to the "input" tuple

$$t^i = (t_1, t_2, ..., t_i)$$

and calls itself recursively with

$$t^{i+1} = (t_1, t_2, ..., f_j)$$

as an input.

The described algorithm can generate elements of set *T* randomly: for that we just need to do a recursive algorithm call at level i not for all $j \in J_k$, but only for q_i % of them (selected randomly). It should be noted that we can use various mechanisms of random selection (in this work we use the simplest way of selection: we just generate a uniformly distributed random value; though other distributions can also be used).

Let us consider specific features of the construction of set F^i for some classes of combinatorial sets. In the case of arrangements with repetitions, set F^i consists of all generating elements:

$$F^i = A.$$

For arrangements without repetitions and permutations as their special case, set F^i consists of $n-i$ generating elements that have not appeared in t^i:

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

As for the combinations the order of the items is not important, let us generate them in the form of ordered sets where $t_1 < t_2 < ... < t_i$ is for combinations without repetitions and $t_1 \le t_2 \le ... \le t_i$ is for combinations with repetitions.

For combinations without repetitions, set F^i includes all generating elements that have not been included into t^i and are greater than t_i:

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

For combinations with repetitions, set F^i also includes generating element which is equal to t_i:

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

Let us describe the algorithm **GenBase** that implements the steps described. **GenBase** input data consist of T - type of a basic set, a set of parameters $p(T)$ and tuple t^i. At each recursion level $i \in J^0_{m-1}$, the algorithm builds the set F^i, then randomly adds selected element F^i to tuple t^i as many as v_i times and recursively calls itself. At each level, the number of recursive calls is determined by

$$v_i = \frac{|F^i| \cdot q_i}{100}.$$

To generate all the elements of set T, **GenBase** should be called with parameters T, $(\ )$, $p(T)$.

Here *random*$(1, |F^i|)$ is a uniformly distributed random number between 1 and $|F^i|$. In order to generate combinatorial sets of other classes with this algorithm (Fig. 1), it is sufficient to identify laws of constructing set F^i.

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**

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

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

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

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

T_n : **exit;**

end case;

$$v_i = \frac{|F^i| \cdot q_i}{100};$$

for $j = 1, 2, ..., v_i$ **do**

GenBase($t^{i+1} = (t_1, t_2, ..., t_i, f_{random(1,|F^i|)})$);

end for;
end function;

Fig. 1. GenBase algorithm

It should be noted that, to generate combinatorial sets of other classes with this algorithm, it is sufficient to identify laws of constructing set F^i.

Example 1. Let the task is to randomly generate arrangements of 4 to 2. Let $q_0 = 75,\ q_1 = 60$. At zero level, set F^0 includes all generating elements: $F^0 = \{1, 2, 3, 4\}$. Hence, $v_0 = \frac{4 \cdot 75}{100} = 3$. The algorithm randomly selects three elements F^0 (let them be 3, 2 and 4) and makes recursive calls of itself with $t^1 = (3),\ t^1 = (2),\ t^1 = (4)$ respectively.

Let us consider the situation at level 1 for $t^1 = (3)$. In this case, set F^1 includes all elements that have not been not included into t^1, i.e. $F^1 = \{1, 2, 4\}$. Hence, $v_1 = \frac{3 \cdot 60}{100} \approx 2$. The algorithm randomly selects two elements F^1 (let them be 1 and 4); it means that, at level 2, full arrangements (31) and (34) are obtained.

A possible recursion tree of the algorithm operation is provided below (Fig. 2).

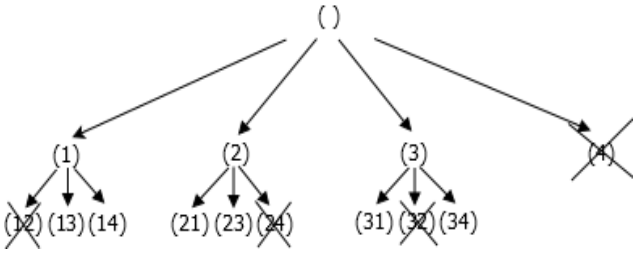


Fig. 2. Example results of **GenBase** algorithm

GENERATION OF *K*-SETS

First, let us briefly describe the algorithm of exhaustive generating *k*-sets [9]. At the beginning, it generates the elements of each basic set using **GenBase** algorithm (values of q_i are individual for each basic set). After that, the algorithm sequentially implements mappings

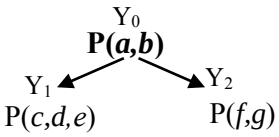
$$\Gamma_{i+1} \circ \Gamma_i \circ .. \circ \Gamma_0(z),\ z = A_0 \in p(Y_0)$$

for each $i \in J_{k-1}^0$. In other words, it implements n-composition, where generating elements of a parent set are replaced by the elements-tuples of its child sets.

Algorithm Get_k-set in its original version in [9], while implementing an operation of *n*-composition, performs a sequential substitution of the element of a parent set by each element of a child set. The proposed algorithm Get_Random_k-set does it not for all, but for some elements of a child set. By analogy with the generation of basic sets, we can put the parameter *Q(Y)* for each basic set *Y* at level $i \in J_k$, which defines the fraction of elements participating in the operation of n-composition. The number of the elements is determined by

$$V(Y) = \frac{|Y| \cdot Q(Y)}{100}.$$

Example 2. Let us describe the generation of the composition of permutations by means of **Get_Random_k-set**. The structure of the corresponding *k*-set can be presented as follows:



The elements "*a*" and "*b*" of set $Y_0 = \{(ab), (ba)\}$ are replaced by the elements of child sets:

$$Y_1 = \{(cde), (ced), (dce),\ (dec), (ecd), (edc)\}$$

and

$$Y_2 = \{(fg), (gf)\},$$

respectively.

Let

$$Q(Y_1) = Q(Y_2) = 50.$$

Hence, $V(Y_1) = 3;\ V(Y_2) = 1$, i.e. element "*a*" is sequentially replaced by three random elements of Y_1, and "*b*" – by one random element of Y_2.

A possible results of the generation is

$$W_z = \{\underbrace{(cdefg),(dcefg),(ecdfg)}_{result\ of\ replacing\ (ab)},$$
$$\underbrace{(gfced),(gfdec),(gfedc)}_{result\ of\ replacing\ (ba)}\}.$$

THE EVALUATION OF THE COMPLEXITY OF ALGORITHM GET_RANDOM_K-SET

The complexity of the algorithm is determined by the complexity of generation of basic sets, as well as the complexity of operations of *n*-substitution and a number of levels of a certain *k*-set. In [9], the evaluation of the complexity of the «full» algorithm **Gen_k-set** is described. This formula can also be used in this case. It is sufficient to replace cardinality *Card(Y)* of each basic set by *V(Y)*. Then, the formula to evaluate **Get_Random_k-set** complexity is:

$$\sum_{i=0}^{k}\sum_{j=1}^{\eta_i} O(Y_{ij}) + \sum_{i=0}^{k-1}(Card(P^i)\prod_{j=1}^{\eta_{i+1}} V(Y_{(i+1)j})),$$

where: $O(Y_{ij})$ is the complexity of generating a basic set Y_{ij} ($\boldsymbol{i}$ is the level of the basic set in *k*-set, $\boldsymbol{j}$ is the sequence number of the set at level $\boldsymbol{i}$); $P^i = \Gamma_i \circ \Gamma_{i-1} \circ ... \circ \Gamma_0(z)$ – "intermediate" *k*-set, which is the parent set at level $\boldsymbol{i}$.

Details about designations and obtaining this formula are given in [9].

CONCLUSIONS

1. We described the general approach for solving the problem of random generation of $k-$sets based on the single approach to random generation of various basic combinatorial sets.

Algorithm for random generation of basic sets allows generating various combinatorial sets, and laws of constructing sets F^i can be pre-set. If identification of the laws fails, the algorithm allows using other algorithms to generate basic sets.

2. The described approach to the random generation is very flexible since it allows obtaining various results by varying parameters q_i and *Q(Y)*. In its turn, this allows adjusting the number of elements for both basic sets and *k*-sets.

3. The developed software allows solving the described problems of generation of *k* -sets and basic sets.

REFERENCES

1. Knuth D., 2005. The Art of Computer Programming, Volume 4, Fascicle 2: Generating All Tuples and Permutations. Addison-Wesley. – 144.

2. Knuth D., 2005. The Art of Computer Programming, Volume 4, Fascicle 3: Generating All Combinations and Partitions. Addison-Wesley.–160.

3. Kreher D., Stinson D., 1999. Combinatorial Algorithms: Generation Enumeration and Search. CRC Press. – 329.

4. Bona M., 2004. Combinatorics of Permutations. Chapman Hall-CRC. – 383.

5. Ruskey F., 2009. Combinatorial generation, Dept. of Computer Science Univ. of Victoria, Canada, 1j-CSC 425/520. – 289.

6. Korsh J., LaFollette P., 2004. Loopless array generation of multiset permutations // The Computer Journal. – Vol. 47, № 5. – 612–621.

7. Stoyan Y., Grebennik I., 2008. Description of Classes of Combinatorial Configurations by Mappings // Proc. of NAS of Ukraine.– 10. - 28-31. (in Russian).

8. Stoyan Y., Grebennik I., 2011. Description and Generation of Combinatorial Sets Having Special Characteristics // International Journal of Biomedical Soft Computing and Human Sciences. Special Volume "Bilevel Programming, Optimization Methods, and Applications to Economics" Vol. 18, №1. – 85-90.

9. Grebennik I., Lytvynenko O., 2012. Generating combinatorial sets with given properties // Cybernetics and Systems Analysis, 48(6). – 890–898.

10. Grebennik I., 2010. Description and generation of permutations containing cycles // Cybernetics and Systems Analysis, 46(6). – 945-952.

11. Grebennik I., Chorna O., 2015. Elements transpositions and their impact on the cyclic structure of permutations. // ECONTECHMOD. An International Quarterly Journal on Economics of Technology and Modelling Processes. – Vol. 4, № 3. – 33–38.

12. Grebennik I., Baranov A., Chorna O., Gorbacheva E., 2016. Optimization of linear function of a cyclic permutation based on the random search. // ECONTECHMOD. An International Quarterly Journal on Economics of Technology and Modelling Processes. – Vol. 5, № 3. – 211–216.

13. Bauslaugh B., Ruskey F., 1990. Generating alternating permutations lexicographically // BIT Numerical Mathematics. 30, 17–26.

14. Poneti M., Vajnovszki V., 2010. Generating restricted classes of involutions, Bell and Stirling permutations // European Journal of Combinatorics. 31, 553–564.

15. Bergeron F., Labelle G., Leroux P., 1998. Combinatorial Species and Tree-Like Structures. University Press, Cambridge.

16. De Bruijn N., 1970. Permutations with given ups and downs // Nieuw Arch. Wisk. 18 (3), 61–65.

17. Carlitz L., 1973. Permutations with prescribed pattern // Math Hachr, vol. 58, No. 1-6, 31–53.

18. Viennot G., 1979. Permutations ayant une forme donne // Discrete Mathematics. 26, 279–284.

19. Etienne G., 1984. Linear extensions of finite posets and a conjecture of G. Kreweras on permutations // Discrete Mathematics. 52, 107–112.

20. Shevelev V., 2012. The number of permutations with prescribed Up-down structure as a function of two variables // Integers. Volume 12.

21. Van Baronaigien R., Ruskey F., Ruskey D., 1992. Generating permutations with given ups and downs // Discrete Applied Mathematics. 36, 57–65.