Algorithms for ranking and unranking the combinatorial set of closed questionnaire answers

P P Shcheglov, G A Filippov, Y V Shablya and D V Kruchinin
Tomsk State University of Control Systems and Radioelectronics, Tomsk, 634050 Russia
E-mail: syv@keva.tusur.ru

Abstract. In this paper, we study the combinatorial set of closed questionnaire answers with a specified minimum number of correct answers. For this combinatorial set, we obtain an explicit formula for its cardinality function. Using the obtained cardinality function, we construct the corresponding AND/OR tree structure and determine the bijection rules for this AND/OR tree and the considered combinatorial set. In addition, we develop ranking and unranking algorithms for the combinatorial set.

1. Introduction
Opened and closed questionnaires are applied to conduct majority tests and surveys. Opened questionnaires include questions that require making your own answer options, whereas closed questionnaires can be answered by picking up an option from a limited list of ones (for example, options “yes” or “no”). Closed questionnaires are commonly used by different organizations such as business, educational, public, and governmental ones. The main purpose of such questionnaires is to monitor or assess knowledge, skills, and other characteristics at the different stages of studying. Due to the fixed number of possible options to answer a closed questionnaire, it becomes possible to automate the processes of storing, transmitting and processing such data [1, 2]. However, if we process big data, then we are faced with the problem of encoding and storing such data.

This kind of problem falls within the scope of the combinatorial generation [3, 4]. The following list of tasks is distinguished in the combinatorial generation:

- Listing: generating elements of a given combinatorial set sequentially;
- Ranking: ranking (numbering) elements of a given combinatorial set;
- Unranking: generating elements of a given combinatorial set in accordance with their ranks;
- Random selection: generating elements of a given combinatorial set in random order.

A combinatorial set is a finite set whose elements have some structure and there is an algorithm for constructing the elements of this set. A combinatorial object is just the particular element of a given combinatorial set.

In this paper, we consider the representation of the set of closed questionnaire answers as a combinatorial set. For this combinatorial set, we develop combinatorial generation algorithms that can be used for encoding its elements.
2. Research methods
There are several general methods for developing combinatorial generation algorithms (such as backtracking [5] or ECO-method [6, 7]). In this paper, we use the method for developing combinatorial generation algorithms that is based on using an AND/OR tree structure [8, 9]. The advantages of this method over the others are the following:

- an ability to develop algorithms for listing, ranking, and unranking combinatorial sets;
- an ability to apply the method for combinatorial sets described by several parameters.

Also, this method has one additional requirement: we need to know the cardinality function of a combinatorial set that belongs to the algebra \( \{ \mathbb{N}, +, \times, R \} \), where \( R \) denotes a primitive recursive function.

An AND/OR tree is a tree structure that contains nodes of two types: AND nodes (a node whose edges to its sons are connected by a curve) and OR nodes (a node whose edges to its sons are not connected by a curve). Also, the recursive nature of the tree structure is represented by nodes with triangles. A variant of an AND/OR tree is a tree structure obtained by removing all edges except one for each OR node. If we know the cardinality function of a combinatorial set that belongs to the algebra \( \{ \mathbb{N}, +, \times, R \} \), then we can construct an AND/OR tree structure for which the total number of its variants is equal to the value of the cardinality function.

3. Main results

3.1. Combinatorial set
Let us consider the following combinatorial object: a sequence of at least \( t \) correct answers to a closed questionnaire, which contains \( n \) questions with \( m \) options to answer each of them and only one option is the correct answer to a question. Figure 1 shows an example of the graphical representation of a closed questionnaire.

![Figure 1. An example of the graphical representation of a closed questionnaire.](image)

For the considered combinatorial set of closed questionnaire answers, we obtain the following explicit formula for the cardinality function:

\[
A(n, m, t) = \sum_{k=t}^{n} C_n^k (m-1)^{n-k},
\]

where \( C_n^k \) is the number of \( k \)-combinations of \( n \) elements [10].

An example of obtaining formulas for enumerating combinatorial sets can be found in [11].
Each element of the combinatorial set (a sequence of answers to a closed questionnaire) is encoded by a sequence \(a = (a_1, a_2, \ldots, a_n)\) of given answers to \(n\) questions, where \(a_i \in \{0, 1, \ldots, m-1\}\). The case of giving the correct answer to the \(i\)-th question corresponds to \(a_i = 0\). Otherwise, in the case of giving an incorrect answer to the \(i\)-th question, the value of \(a_i\) shows the position of the chosen answer among \(m-1\) incorrect answers.

Using Equation (1), we can calculate the total number of sequences of answers to a closed questionnaire that satisfy the above mentioned requirements. In this formula, \(k\) shows the number of correct answers to \(n\) questions. Based on the description of the combinatorial object, this value should be no less than \(t\), i.e. \(t \leq k \leq n\). Then, for a fixed value of \(k\), we need to determine which \(k\) questions were answered correctly. The number of possible ways to select \(k\) questions from \(n\) ones is determined by the number of \(k\)-combinations of \(n\) elements. The remaining \(n-k\) questions must be answered incorrectly. The number of possible ways to answer incorrectly is equal to \(m-1\), because only one option from \(m\) options is the correct answer to a question.

The number of \(k\)-combinations of \(n\) elements can be calculated by the following recurrence relation [10]:

\[
C_n^k = C_{n-1}^k + C_{n-1}^{k-1}, \quad C_n^n = C_n^0 = 1.
\]

Since Equation (1) and Equation (2) belong to the algebra \(\{\mathbb{N}, +, \times, R\}\), then we can construct the AND/OR tree structure for \(A(n, m, t)\) that is presented in Figure 2.

![Figure 2. An AND/OR tree for \(A(n, m, t)\).](image)

For a compact representation, we encode a variant of the AND/OR tree by a sequence \(v = (k, vl, vr)\), where:

- \(k\) corresponds to the label of the selected son of the OR node labeled \(A(n, m, t)\);
- \(vl = (v_1, v_2, \ldots)\) corresponds to the variant of the subtree of the node labeled \(C_n^k\) (this variant is represented as a sequence of the selected sons in the subtree where \(v_i = 0\) corresponds to the left son and \(v_i = 1\) corresponds to the right son);
- \(vr = (v_1, v_2, \ldots, v_{n-k})\) corresponds to the variant of the subtree of the node labeled \(n-k\) (this variant is represented as a sequence of the selected leaves in the subtree where \(v_i \in \{1, \ldots, m-1\}\)).
Figure 3 presents an example of a variant of the AND/OR tree for $A(n,m,t)$ where $n = 4, m = 5$ and $t = 2$. According the above mentioned rules, this variant can be encoded by a sequence $v = (2, (0,1,0), (3,2))$.

![Figure 3. A variant of the AND/OR tree for $A(4,5,2)$.

3.2. Bijection between the AND/OR tree and the combinatorial set
The next step in developing combinatorial generation algorithms is obtaining a bijection between the elements of the combinatorial set and the set of variants of the AND/OR tree. To determine the bijection rules, we look at the changes that take place in going through the whole tree:

- the number of correct answers to $n$ questions is determined by the value of $k$ that corresponds to the label of the selected son of the OR node labeled $A(n,m,t)$ in a variant of the AND/OR tree (the value of $k$ also shows the number of zeros in the sequence $a = (a_1,a_2,\ldots,a_n)$);
- the left son of the AND node that is labeled by $C_n^k$ determines $k$ questions from $n$ ones that were answered correctly (for this subtree, each selected left son labeled $C_n^{k-1}$ determines that the first question was answered incorrectly and each selected right son labeled $C_n^{k-1}$ determines that this question was answered correctly, and these rules are repeated recursively for the next questions);
- the right son of the AND node that is labeled by $n-k$ determines options that were chosen as incorrect answers for the remaining $n-k$ questions (there are $m-1$ such options for each of $n-k$ questions).

Using the obtained bijection rules, we can discover the corresponding combinatorial object $a = (a_1,a_2,a_3,a_4)$ for the presented variant $v = (2, (0,1,0), (3,2))$ in Figure 3:

1. The first value 2 shows the number of correct answers, i.e. there are 2 zeros among $a_i$.
2. The sequence $(0,1,0)$ shows that:
   - the first question was answered incorrectly ($v_1 = 0$), i.e. $a_1 \neq 0$;
   - the second question was answered correctly ($v_2 = 1$), i.e. $a_2 = 0$;
   - the third question was answered incorrectly ($v_3 = 0$), i.e. $a_3 \neq 0$;
   - there is only one question left (the forth question) and it remains to choose one question that must be answered correctly, i.e. $a_4 = 0$. 

3. The sequence $(3, 2)$ shows that:

- for the first question that was answered incorrectly ($a_1$), it is necessary to choose the third incorrect option ($v_1 = 3$), i.e., $a_1 = 3$
- for the second question that was answered incorrectly ($a_3$), it is necessary to choose the second incorrect option ($v_2 = 2$), i.e., $a_3 = 2$

Hence, we can get that the variant $v = (2, (0,1, 0), (3, 2))$ corresponds to the object $a = (3, 0, 2, 0)$.

3.3. Bijection between the AND/OR tree and the combinatorial set

Using the method for developing combinatorial generation algorithms based on AND/OR trees, we can develop algorithms for ranking (Algorithm 1) and unranking (Algorithm 2) the variants of the constructed AND/OR tree for $A(n,m,t)$.

**Algorithm 1:** An algorithm for ranking the variants of the AND/OR tree for $A(n,m,t)$.

```plaintext
1  RankVariant_A (v = (k, vl, vr), n, m, t)
2   begin
3      if k = n then r := \sum_{i=0}^{k-1} C_n^i (m-1)^{n-i}
4      else
5         l_1 := RankVariant_C (vl, n, k)
6         l_2 := vr_{n-k} - 1
7         for i := n-k-1 to 1 do l_3 := vr_{i-1} + (m-1)l_2
8            r := l_1 + C_n^i l_3 + \sum_{i=0}^{k-1} C_n^i (m-1)^{n-i}
9      end
10     return r
11  end
```

In these algorithms, we used algorithms for ranking and unranking the variants of the AND/OR tree for $C_n^k$ [9]. For example, if we consider the variant $v = (2, (0,1, 0), (3, 2))$ and apply the developed Algorithm 2, then we can get the corresponding rank $r = 37$.

The developed algorithms depend on calculating the value of the cardinality function $A(n,m,t)$ and have polynomial time complexity $O((n-t)^2)$. That is, in the summation we need to add $(n-t)$ times the product of $(n-t)$ numbers (raising $(m-1)$ to the $(n-t)$-th power).

4. Conclusion

In this paper, we study the combinatorial set of closed questionnaire answers and obtain an explicit formula for the cardinality function $A(n,m,t)$ of this combinatorial set that belongs to the algebra $\{\mathbb{N}, +, \times, R\}$. Using the obtained cardinality function, we construct the corresponding AND/OR tree structure. For the set of all variants of this AND/OR tree structure, we determine the bijection rules that allow us to find the corresponding combinatorial object for each variant. Using the method for developing combinatorial generation algorithms based on AND/OR trees, we develop algorithms for ranking and unranking the variants of the constructed AND/OR tree for $A(n,m,t)$. 

Algorithm 2: An algorithm for unranking the variants of the AND/OR tree for \( A(n,m,t) \).

\[
\text{UnrankVariant}_A \ (r, n, m, t) \begin{align*}
1 & \quad \text{begin} \\
2 & \quad \quad k := t \\
3 & \quad \quad \text{sum} := 0 \\
4 & \quad \quad \textbf{while} \quad \text{sum} + C_n^k (m-1)^{n-k} \leq r \\
5 & \quad \quad \quad \text{sum} := \text{sum} + C_n^k (m-1)^{n-k} \\
6 & \quad \quad \quad k := k + 1 \\
7 & \quad \quad \textbf{end} \\
8 & \quad r := r - \text{sum} \\
9 & \quad l_1 := r \mod C_n^k \\
10 & \quad l_2 := \left\lfloor \frac{r}{C_n^k} \right\rfloor \\
11 & \quad v \leftarrow \text{UnrankVariant}_C \ (l_1, n, k) \\
12 & \quad \textbf{for} \ i := 1 \ \textbf{to} \ n-k \ \textbf{do} \\
13 & \quad \quad v_r := (l_2 \mod (m-1)) + 1 \\
14 & \quad \quad l_2 := \left\lfloor \frac{l_2}{(m-1)} \right\rfloor \\
15 & \quad \textbf{end} \\
16 & \quad v = (k, v_l, v_r) \\
17 & \quad \textbf{return} \ v \\
\end{align*}
\]

Acknowledgments
The reported study was supported by the Russian Science Foundation (project no. 18-71-00059).

References
[1] Desai S C and Reimers S 2019 Comparing the use of open and closed questions for web-based measures of the continued-influence effect Behavior Research Methods 51 1426–1440
[2] Reja U, Manfreda K L, Hlebec V and Vehovar V 2003 Open-ended vs. close-ended questions in web questionnaires Developments in Applied Statistics 19 159–177
[3] Knuth D E 2011 The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1 (USA: Addison-Wesley)
[4] Ruskey F 2003 Combinatorial Generation. Working version (1j-CSC 425/520) Available at http://page.math.tu-berlin.de/felsner/SemWS17-18/Ruskey-Comb-Gen.pdf
[5] Kreher D L and Stinson D R Combinatorial Algorithms: Generation, Enumeration, and Search (USA: CRC Press)
[6] Barcucci E, Del Lungo A, Pergola E and Pinzani R 1999 ECO: A methodology for the enumeration of combinatorial objects J. Difference Equ. Appl. 5 435–490
[7] Bacchelli S, Barcucci E, Grazzini E and Pergola E 2004 Exhaustive generation of combinatorial objects by ECO Acta Inform. 40 585–602
[8] Shablya Y V and Kruchinin D V 2019 Modification of the algorithm development method for combinatorial generation based on the application of the generating functions theory Proceedings of TUSUR 22 55–60
[9] Shablya Y V, Kruchinin D V and Kruchinin V V 2020 Method for developing combinatorial generation algorithms based on AND/OR trees and its application Mathematics 8 962
[10] Graham R L, Knuth D E and Patashnik O 1994 Concrete Mathematics. Second ed. (USA: Addison-Wesley)
[11] Banderier C, Krattenthaler C, Krinik A, Kruchinin D, Kruchinin V, Nguyen D and Wallner M 2019 Explicit formulas for enumeration of lattice paths: Basketball and the kernel method Lattice Path Combinatorics and Applications (Springer, Cham) pp 78–118