Advances and Applications in Discrete Mathematics

http://www.pphmj.com
http://dx.doi.org/10.17654/0974165822039
Volume 34, 2022, Pages 1-15 **P-ISSN: 0974-1658**

GENERALIZED PASCAL'S PYRAMIDS AND DECISION TREES

O. V. Kuzmin

Department of Probability Theory and Discrete Mathematics
Irkutsk State University
Irkutsk, Russia
e-mail: quzminov@mail.ru

Abstract

This work studies combinatorial objects of pyramidal structure. It considers one of the ways of representing rules in a hierarchical consecutive structure: the decision tree method, where each object corresponds to the single node that provides a solution. The paper proposes algorithms of building a decision tree based on the generalized Pascal's triangle and generalized Pascal's pyramid.

The offered methods of the combinatorial analysis of hierarchical structures in decision-making problems can be used in creating and analyzing knowledge databases.

Received: July 5, 2022; Accepted: September 3, 2022
2020 Mathematics Subject Classification: 06E30, 94D10, 15B34, 05B20.
Keywords and phrases: hierarchical structure, partially ordered set, generalized Pascal's pyramid, decision-making problems, decision tree, combinatorial algorithms.

How to cite this article: O. V. Kuzmin, Generalized Pascal's pyramids and decision trees, Advances and Applications in Discrete Mathematics 34 (2022), 1-15.
http://dx.doi.org/10.17654/0974165822039

Published Online: September 27, 2022

0. Introduction

Combinatorial problems of algorithmic nature on discrete finite mathematic structures are encountered in practice. In recent years, interest in the theory of "large systems", which we have to face in widely differing fields of science and technology, has been on the rise. An important research trend of these "large" or "complex" systems is to consider them as multilevel or hierarchically structured [1, 2]. The process of the stage-by-stage building of a solution of multicriterial problems with hierarchical structures can often be interpreted as a trajectory on a finite lattice [3, 4], describing the corresponding partially ordered set [5]. Similar problems are not uncommon in developing methods of automatic analysis of bulk data in information systems and network processing [6, 7].

Monograph [8] presents a scheme of constructing combinatorial numbers and polynoms based on a hierarchical pyramidal structure with weights, which is called the *generalized Pascal's pyramid*. In [9], a widely known technique of the theory of partially ordered Roth-Stanley's sets [3] is applied to research a whole host of combinatorial objects described by scheme [8].

Decision trees are one of the most effective instruments of intellectual data analysis and predictive analytics that make it possible to solve classification and regression problems.

The essential ideas that led to the emergence and development of decision trees were planted in the 1950s in the field of a computer-aided research of human behavior modeling. Among them, one should emphasize the works of Hovland [10] and Hunt et al. [11].

The further development of decision trees as self-correcting models for data analysis is associated with Quinlan [12, 13] who developed algorithm ID3 and its advanced versions C4.5 and C5.0, and with Breiman et al. [14] who proposed the CART algorithm and the random forest method.

This work relates to the field of development of methods of analysis of hierarchical systems and their applications in decision-making problems. It

develops algorithms of constructing a hierarchical classification model - the decision-making model.

Section 1 introduces basic concepts and fundamental relations for the studied combinatorial objects. Among the latters are the generalizations of Pascal's triangle and pyramid with multiple applications.

Section 2 considers one of the ways of representing rules in a hierarchical consecutive structure: a decision tree method, where each object corresponds to the single node that provides a solution. The work proposes algorithms of building decision trees based on the generalized Pascal's triangle and generalized Pascal's pyramid.

Section 3 presents a basic principle on which the combinatorial search methods by means of the decision tree are based. It consists in decomposing the initial problem into some number of subproblems with a subsequent attempt to solve each of them. Search methods using the decision trees without repeating situations are considered.

Finally, Section 4 presents conclusion.

1. Basic Concepts and Relations

The generalized Pascal's pyramid (or *V-pyramid*) [8] is a hierarchical trihedral pyramidal structure V with weights, the elements of which satisfy recurrent relations

$$V(n,\,k,\,l) = \alpha_{n,\,k-1,\,l}\,V(n-1,\,k-1,\,l) + \beta_{n,\,k,\,l-1}\,V(n-1,\,k,\,l-1) + \gamma_{n,\,k,\,l}\,V(n-1,\,k,\,l) \qquad (1)$$

with boundary conditions

$$V(0,\,0,\,0) = V_0, \quad V(n,\,k,\,l) = 0,$$

if $\min(n,\,k,\,l,\,n-k-l) < 0$.

Number V_0, standing in the vertex (zero layer) of the generalized Pascal's pyramid, is addressed specifically. In many cases, one can assume that $V_0 = 1$. Quantities $\alpha_{n,k,l}$, $\beta_{n,k,l}$, $\gamma_{n,k,l}$ are called *weight factors* or *weights*. A number of properties of *V*-pyramids and their special cases are presented in works [15-18].

The upper N-unit of the *V*-pyramid (V_N *-pyramid*) as a result of clipping is a finite trihedral pyramidal array of elements that coincides with the upper part of the *V*-pyramid [17]. With that, a plane *N*-section of the *V*-pyramid acts as the foundation of the V_N -pyramid, and the numbers of the considered V_N -pyramids coincide with the numbers of the corresponding sections of the *V*-pyramid.

An important special case of the generalized Pascal's pyramid is the *generalized Pascal's triangle* (or *V*-triangle) [8] defined as a hierarchical triangular structure V with weights, whose elements satisfy recurrent relations

$$V(n,\, k) = \alpha_{n,k-1}\, V(n-1,\, k-1) + \beta_{n,k}\, V(n-1,\, k) \qquad (2)$$

with boundary conditions $V(0,\, 0) = V_0$, $V(n,\, k) = 0$, if $\min(n,\, k,\, n-k) < 0$. Figure 1 shows a hierarchical structure V with weights which is described by relation (2), denoting $V_{nk} = V(n,\, k)$ for brevity.

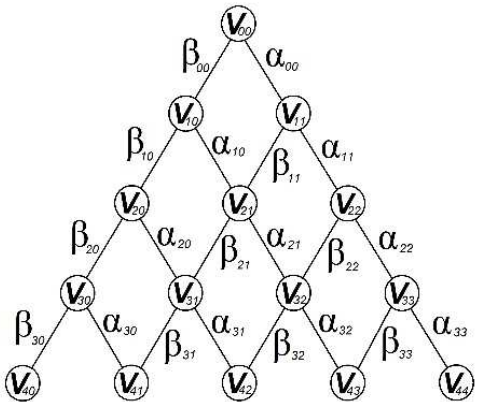

Figure 1. The hierarchical structure of the *V*-triangle.

When preassigning respective sets of values of weights, relations (1) and (2) allow enumerative interpretations in terms of lattice paths which, in turn, can be used in the lattice structure analysis [19, 20].

Further, we need the following *graph operations* [21].

The *removal of the vertex* $v \in V$ of graph $G = G(V, E)$ is an operation that consists in removing this vertex together with all the edges incident with it. To designate a graph obtained after removing the vertex v of the graph G, a symbol $G\backslash v$ is used.

The removal of the edge $e \in E$ of the graph $G = G(V, E)$ is an operation that consists in removing this edge while maintaining all the graph vertices. To designate a graph obtained after removing the edge e of the graph G, a symbol $G\backslash e$ is used.

The contraction of the edge $e = (u, v)$ of the graph $G(V, E)$ is an operation that consists in identifying adjacent vertices u and v and removing the formed loop.

It should be reminded [22] that the *height* of a rooted tree is the maximal number of arcs that separate all the leaves from the root. If the tree is not weighted, then its height is just the distance from the root to the outermost leaf.

In applications, it is not uncommon to deal with *binary decision trees*. In this case, the root and each inner vertex have no more than two descendants. At the level h of the binary tree, there are at most 2^h vertices. A binary tree of height h that has the maximal number of nodes is a *perfect* binary tree.

Ternary trees, in which the root and each inner vertex have no more than three descendants, are also applied. The common applications for search ternary trees include spell checking and auto complete functions.

2. Building Decision Trees

A model of a multi-stage decision-making problem that is considered below implies the presence of a certain graph describing the way it is possible to get from the preassigned set of its initial vertices to the preassigned set of its terminal vertices (that is, vertices of degree 1).

The following method applied for decision-making in risky conditions is called a *decision tree*. It is applied when it is necessary to make a sequence of decisions. The method derived its name from its tree-type structure consisting of the decision rules of an "if...., then ...." type and making it possible to classify objects. Building the decision tree starts from an earlier decision, then possible actions and consequences of each action (event) are indicated. Thereafter, a decision is made again (action direction choice) and further until such time as all the logical consequences of the results are settled.

The decision tree is a way of organizing data as a hierarchical structure that includes two types of elements: *nodes*, or *inner* vertices, and *leaves*, or *hanging* vertices that are different from the *root*. A leaf determines a decision for each example that falls within it. It does not contain a rule, but a subset of objects satisfying all the rules of the branch that ends with this leaf. There is only one path to each leaf, that is why an example or an assertion can fall within one leaf only. This assures the uniqueness of the decision.

At present, decision trees are one of the most popular *data mining* methods used in solving classification problems.

Let us pass on to the description of a developed algorithm of building decision trees based on the generalized Pascal's triangle and generalized Pascal's pyramid.

Algorithm 1.

BEGIN

Step 1. We determine input parameters for finding dimensions of V_N - the upper *N*-unit of the *V*-pyramid.

Step 2. We determine parameters for building a set of forbidden vertices $V' \subset V$.

Step 3. We determine parameters for building a set of forbidden edges E'.

Step 4. We determine a set of forbidden vertices $V' \subset V$.

Step 5. We determine a set of forbidden edges $E' \subset E$.

Step 6. We remove from the set V vertices belonging to V', that is, replace V with $V \backslash V'$.

Step 7. We remove from the set of edges E edges belonging to $E'' \subset E'$, that is, forbidden edges left after performing Step 5.

Step 8. If we find three vertices u, w, v that form (respectively, counting from the root) a (simple) chain, provided that $\deg(w) = 2$, then we pass on to Step 9, otherwise it is Step 10.

Step 9. We contract the edge (w, v) to the vertex v and go on to Step 7.

Step 10. We assume that all the edges left in V_N have weights equal to 1.

Step 11. Tree *D* is built.

END

Algorithm 1 can be particularly used in building a ternary tree *T*.

In constructing binary trees, we can use a special case of Algorithm 1 - Algorithm 2, in which Step 1 is substituted for the following:

Step 1′. We determine input parameters for finding dimensions of V_N - the upper *N*-unit of the *V*-triangle.

To determine input parameters for identifying dimensions of V_N - the upper *N*-unit of the *V*-triangle, one can use the following assertions:

Assertion 1. To build a perfect binary tree $\mathcal{B}$ with root V_0 of height *n*, a V_N-triangle, where $N = 2^n$, is necessary and sufficient.

Proof. Let us prove it by a method of complete mathematical induction.

It is assumed that tree $\mathcal{B}$, consisting solely of the root, has a height of 0. With $n = 0$ and $n = 1$, we correspondingly obtain $N = 1$ and $N = 2$, and the assertion is checked immediately.

Let the assertion be valid for some fixed $N = 2^n$. Since at the level *n* of the binary tree $\mathcal{B}$, there are at most 2^n vertices, the corresponding upper N_1-unit of the *V*-triangle must have rows with numbers up to and including 2^{n+1}.

To prove the sufficiency, let us consider rows of the *V*-triangle with numbers from $2^n + 1$ to 2^{n+1} inclusive. There are two edges stemming from each vertex of level *N* (from right to left). From these edges, two (different) paths, respectively, start to two neighboring vertices of a layer with number $N_1 = 2^{n+1}$. In doing so, a set of vertices of level $N + 1$ is divided into 2^n pairs of the neighboring vertices having one of the vertices in the layers from 2^n to $2^{n+1} - 1$ as the common ancestor.

Assertion 2. To build a binary tree $\mathcal{B}$ with root V_0 of height *n*, a V_N-triangle, where $N = 2^n$, is sufficient.

Proof. It follows from Assertion 1 and the fact that any binary tree $\mathcal{B}$ with root V_0 of height *n* is isomorphic to some subtree of the perfect binary tree $\mathcal{B}$.

To show how Algorithm 2 is applied to specific problems, consider an example of building the binary decision tree $\mathcal{B}$ based on the *V*-triangle.

By means of preassigning weights, sets $V' \subset V$ of *forbidden vertices* (*forbidden positions* [3]) and $E' \subset E$ - *forbidden edges*, removing all elements of sets V' and E' and subsequently contracting edges, the hierarchical structure described by relation (2) can be transformed into the corresponding binary decision tree $\mathcal{B}$ (Figures 2 and 3).

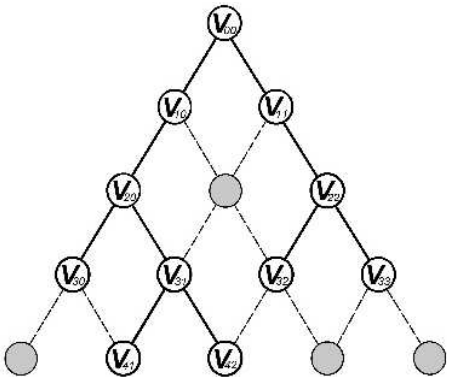

Figure 2. The *V*-triangle with forbidden edges and vertices.

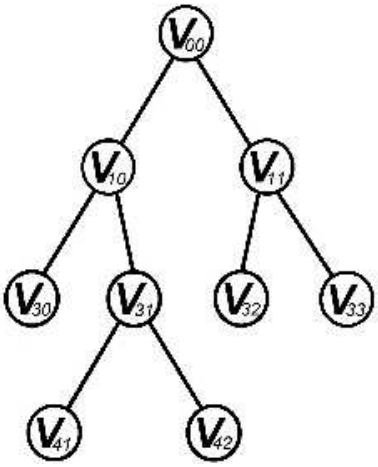

Figure 3. The binary decision tree $\mathcal{B}$.

Figure 2 is obtained from Figure 1 where the forbidden vertices are marked in grey, the forbidden edges – with dashed lines, and the rest of the

weights are assumed to be set to unity. Figure 3 presents the binary decision tree $\mathcal{B}$ obtained from the V-triangle in Figure 2 by removing forbidden vertices from set $V' = \{V_{2,1}, V_{4,0}, V_{4,3}, V_{4,4}\}$, removing edges $(V_{3,0}, V_{4,1})$ and $(V_{3,2}, V_{4,2})$ from E', and subsequently contracting edges $(V_{1,1}, V_{2,0})$ and $(V_{1,0}, V_{2,2})$ in vertices $V_{1,0}$ and $V_{1,1}$, accordingly.

The advantage of the simplest decision trees is their obviousness. They do not deal with probabilities or weights. To solve real problems, complicated and complemented modifications of search trees are used.

Decision tree method is applied in classification and prediction problems, when it is necessary to make decisions under risk and uncertainty, and the outcome of the events depends on probabilities. Each decision is influenced by some certain factors, and each decision has its own consequence of a probabilistic nature.

Note 1. When preassigning respective numbers of values of weights (and taking into consideration normalizing conditions), relations (1) and (2) allow probabilistic interpretations [8] in solving real problems in terms of modified decision trees.

In which case, one can use modifications of the provided algorithms – Algorithm 1* or Algorithm 2*, in which Step 10 is substituted by the following:

Step 10*. For the rest of edges left in V_N, the values of weights are assumed to be equal to the corresponding $p_i > 0$, considering a normalization condition according to the tree levels.

3. Search Methods Using Decision Trees

According to [22], the fundamental principle on which search methods using decision trees are based is the decomposition of the initial problem P_0 into a certain number of subproblems P_1, ..., P_k (that generally represent the whole problem P_0) with an attempt to solve each of them:

(1) to find the optimal solution of the problem P_i, if this problem is so simple that the optimal solution can be obvious;

(2) to show that solving problem P_i has no sense, as a value of the optimal solution for P_i is undoubtedly worse than for the best solution from the ones found earlier;

(3) to show that the subproblem P_i is not permissible.

This division is described by the *tree* shown in Figure 4 where vertices P_1, ..., P_k represent subproblems.

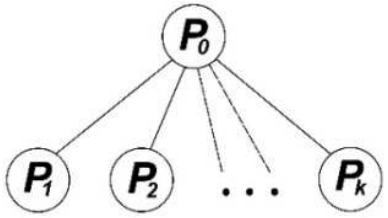

Figure 4. Decomposition of the problem P_0 into subproblems.

The idea of *decomposing* the problem P_0 into some number of subproblems P_1, ..., P_k is that these subproblems are either simpler to solve, or they are smaller, or their structure is not inherent in the original problem P_0. If the subproblem P_i cannot be solved, then it is divided into new subproblems P_{i_1}, P_{i_2}, ..., P_{i_r}, as shown in Figure 5. This division (which is also called *bifurcation*) is repeated for each subproblem which cannot be solved.

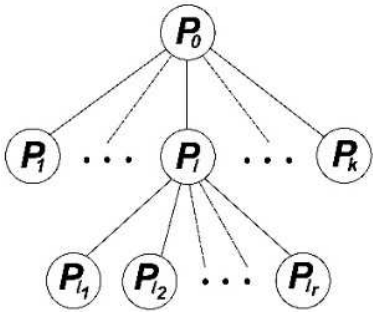

Figure 5. The tree after bifurcation in vertex P_i.

At any stage, the complete set of subproblems that need a solution is presented as a set of *terminal* vertices (that is, vertices of degree 1) of all chains originating from the root (the initial problem P_0) of the search tree. These vertices are called *hanging*. In Figure 5, these are vertices P_1, ..., P_{i-1}, P_{i_1}, P_{i_2}, ..., P_{i_r}, P_{i+1}, ..., P_k.

If the search is exhausted, the set of subproblems into which the problem is divided must represent the entire initial problem. Thus, if the problem P_i is divided into *r* subproblems P_{i_1}, P_{i_2}, ..., P_{i_r}, then

$$\{P_{i_1}\} \cup \{P_{i_2}\} \cup \cdots \cup \{P_{i_r}\} = P_i, \tag{3}$$

where $\{P\}$ determines the set of all permissible solutions of the problem *P*.

Since relation (3) must be applied to each division,

$$\{P_0\} = \bigcup \{P(i) \mid P(j) \text{ is a hanging vertex of the tree}\}. \tag{4}$$

In cases when it is required to enumerate all the solutions of the problem P_0 (and not just find the optimal solution), it is preferable to be able to completely enumerate the solutions using the above-mentioned decomposition of the problem into subproblems and completely enumerate the solutions for each of these subproblems. In this case, it is necessary to avoid duplication of built solutions, that is, one needs to divide the problem P_i into subproblems P_{i_1}, P_{i_2}, ..., P_{i_r} in such a way as to

$$\{P_{i_s}\} \bigcap \{P_{i_q}\} = \varnothing \tag{5}$$

for any two subproblems P_{i_s} and P_{i_q} for which $s \neq q$.

Relation (5) determines its own division of the problem P_i. Though condition (5) is not obligatory for a fully functional search with the decision tree, nevertheless, it is highly profitable from a computational viewpoint, since:

(1) for the P_0 optimization problem, the optimal solution is the solution for one and only one subproblem presented by a hanging vertex;

(2) for the complete enumeration problem, the union of decision sets of subproblems presented by hanging vertices gives the set of all solutions of the problem P_0 without duplication.

Note 2. The decision tree is not normally set a priori when searching. It is built directly in the process of search: when a certain situation emerges, then possible directions for the process are determined. These directions are presented as a set of arcs starting from the vertex that corresponds to the situation. It is natural to reduce the number of these arcs when possible in order to find the solution faster. The methods of reduction are built with consideration to specific problems.

Two main factors that make a binary search tree (often called *BST* for shortness) the optimal solution for any real problems are speed and accuracy. BST-based algorithms are often based on real solutions such as games, data autocomplete and graphics.

4. Conclusion

This work studies combinatorial objects of pyramidal and triangular structures. It considers generalizations of Pascal's triangle and Pascal's pyramid.

The paper proposes algorithms of building the decision tree based on the generalized Pascal's triangle and generalized Pascal's pyramid. Application of the decision tree makes it possible to evidently demonstrate data structure and create a model of data classification, as cumbersome as they can be.

The developed methods of the combinatorial analysis of hierarchical structures indecision-making problems can be used in creating and analyzing knowledge databases.

Acknowledgement

The study was funded by RFBR and the Government of the Irkutsk Region, project number 20-41-385001.

References

[1] M. Mesarović, D. Mako and Y. Takahara, Theory of Hierarchical Multilevel Systems, Academic Press, New York, 1970, 294 pp.

[2] T. L. Saaty, The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation, McGraw-Hill, New York, 1980, 287 pp.

[3] R. P. Stanley, Enumerative Combinatorics, Vol. 1, Cambridge University Press, Cambridge, 1997, 335 pp.

[4] G. Grätzer, Lattice Theory: Foundation, Springer, Basel AG, 2011, 644 pp.

[5] A. A. Balagura and O. V. Kuzmin, Generalized Pascal pyramid and partially ordered sets, Surveys on Applied and Industrial Mathematics 14(1) (2007), 88-91 (in Russian).

[6] V. B. Lebedev and E. A. Fedotov, Information system data modeling using lattice theory methods, University Proceedings, Volga region, Engineering Sciences, 2015, pp. 104-110 (in Russian).

[7] S. K. Murthy, Automatic construction of decision trees from data: a multidisciplinary survey, Data Min. Knowl. Discov. 2(4) (1998), 345-389.

[8] O. V. Kuzmin, Generalized Pascal Pyramids and their Applications, Nauka Publ., Novosibirsk, 2000, 294 pp. (in Russian).

[9] O. V. Kuzmin, A. A. Balagura, V. V. Kuzmina and I. A. Khudonogov, Partially ordered sets and combinatory objects of the pyramidal structure, Advances and Applications in Discrete Mathematics 20(2) (2019), 219-236.

[10] C. I. Hovland, Computer simulation of thinking, American Psychologist 15(11) (1960), 687-693.

[11] E. B. Hunt, Marin Janet and J. S. Philip, Experiments in Induction, Academic Press, New York, 1966, 247 pp.

[12] J. R. Quinlan, Induction of decision trees, Machine Learning 1(1) (1986), 81-106.

[13] J. R. Quinlan, C4.5: Programs for Machine learning, Morgan Kaufmann Publishers Inc., San Mateo, 1993, 302 pp.

[14] L. Breiman, J. Friedman, R. Olshen and C. Stone, Classification and Regression Trees, Wadsworth Books, New York, 1984, 358 pp.

[15] A. A. Balagura and O. V. Kuzmin, Generalized Pascal pyramids and their reciprocals, Discrete Math. Appl. 17(6) (2007), 619-628.

[16] B. A. Bondarenko, Generalized Pascal Triangles and Pyramids, their Fractals, Graphs, and Applications, The Fibonacci Association, Santa Clara, 2010, 296 pp.

[17] O. V. Kuzmin and M. V. Seregina, Upper units of the generalized Pascal pyramid and their interpretations, Journal of Siberian Federal University, Mathematics and Physics 3(4) (2010), 533-543 (in Russian).

[18] O. V. Kuzmin and M. V. Seregina, Plane sections of the generalized Pascal pyramid and their interpretations, Discrete Math. Appl. 20(4) (2010), 377-389.

[19] O. V. Kuzmin, A. P. Khomenko and A. I. Artyunin, Discrete model of static loads distribution management on lattice structures, Advances and Applications in Discrete Mathematics 19(3) (2018), 183-193.

[20] O. V. Kuzmin, A. P. Khomenko and A. I. Artyunin, Development of special mathematical software using combinatorial numbers and lattice structure analysis, Advances and Applications in Discrete Mathematics 19(3) (2018), 229-242.

[21] L. Lovász, Combinatorial Problems and Exercises, Akadémiai Kiadó, Budapest, 1979, 551 pp.

[22] Nicos Christofides, Graph Theory: An Algorithmic Approach, Academic Press, London, 1975, 400 pp.