Method of Constructing Lexicographic Equivalence for Solving Linear Combinatorial Optimization Problems on Arrangements: Results of Computational Experiment

Tetiana Barbolina [0000-0002-4596-7907]

Poltava V.G. Korolenko National Pedagogical University, Ostrogradski str., 2, Poltava, 36003, Ukraine
tm-b@ukr.net

Abstract. The method of constructing lexicographic equivalence for solving linear mixed combinatorial optimization problems on arrangements is considered. The software that implements algorithms of this method is described. The efficiency of algorithms is analyzed be means of computational experiment.

Keywords: Euclidean combinatorial optimization, problems on arrangements, computational experiments, method of constructing lexicographic equivalence.

1 Introduction

Actual trend of the modern theory of optimization is to study the problems of combinatorial nature. In particular, these problems are examined by [1–7]. Important results have been obtained as a result of immersion of combinatorial sets in Euclidean space and study the properties of such problems. Y.G.Stoyan started the theory of Euclidean combinatorial optimization, his disciples – O.O. Iemets, S.V. Yakovlev, I.V. Grebennik and numerous other representatives of this school carry out research in this sphere. They have studied in their works both and properties of Euclidean combinatorial sets immersed in the arithmetical space, and extreme properties of the objective functions, and methods of solving Euclidean combinatorial optimization problems.

This article considers such an important class of Euclidean combinatorial optimization problems as problems on arrangement. The properties of the convex hull of the set of arrangements are explored in [8, 9], in [10, 11] and others — properties of the solution of certain classes of problems on arrangements, a number of results on the solving methods and algorithms of the optimization problems on arrangements in a rather general statement are summarized in [12]. In particular, method of constructing lexicographic equivalence for solving completely linear combinatorial problems on arrangements is substantiated. In [13] this method was extended to mixed combinatorial problems.

The aim of the paper is to describe the software implementation of the algorithms of the method of constructing lexicographic equivalence for solving linear mixed combinatorial optimization problems on arrangements and to present results of computational experiments.

New scientific result obtained in the paper is investigation of an effectiveness of algorithms of method of constructing lexicographic equivalence. Since theoretical estimates are not obtained, then the effectiveness is analyzed by mean of computational experiments.

2 Formal problem statement

Let's consider necessary definitions and facts. As multiset G we understand set of elements, which can include and similar ones. Any multiset $G=\left\{g_1,g_2,...,g_\eta\right\}$ can be assigned by its base $S\left(G\right)$, i.e. the tuple of all its different elements, and by multiplicity – number of repetition of each element of the base. The tuple of multiplicities in the order that corresponds to the elements of the base is called primary specification and is defined by $\left[G\right]$. The set is called the Euclidian combinatorial set, the different elements of which are different ordered k-samples from the multiset $G=\left\{g_1,g_2,...,g_\eta\right\}$ of the representation

$$\left(g_{i_1},g_{i_2},...,g_{i_k}\right), \tag{1}$$

$g_{i_j}\in G$, $i_j\neq i_j$ $\forall i_j,i_t\in J_n$, $\forall j,t\in J_k$ (here and after J_k defines set of k first natural numbers). Examples of Euclidian combinatorial sets are [9] general set of arrangements $E_\eta^k\left(G\right)$ — set of all k-samples of the representation from the multiset G, general multiset of permutations $E_\eta\left(G\right)=E_\eta^\eta\left(G\right)$.

The introduction of the concept of Euclidean combinatorial sets allows highlighting from problems of combinatorial nature the class of problems where the feasible set is Euclidean combinatorial set. In particular, the linear Euclidean problem of combinatorial optimization on arrangements is to find a pair $\left\langle L\left(x^*\right),x^*\right\rangle$ (consisting of maximum and maximal) such that

$$L\left(x^*\right)=\max_{x\in R^n}\sum_{j=1}^{n}c_jx_j\ ,\ x^*=\arg\max_{x\in R^n}\sum_{j=1}^{n}c_jx_j\ , \tag{2}$$

under the combinatorial condition

$$\left(x_1,x_2,...,x_k\right)\in E_\eta^k\left(G\right) \tag{3}$$

and additional linear constraints

$$\sum_{j=1}^{n}a_{ij}x_j\leq b_i,\ i\in J_m\ , \tag{4}$$

where $n \geq k$, $x = (x_1, x_2, ..., x_k) \in R^k$, $c_j \in R^1$ $\forall j \in J_k$, $E^k_\eta(G)$ is a general set of arrangements from the elements of multiset $G = \{g_1, g_2, ..., g_\eta\}$. Variables $x_1, x_2, ..., x_k$ are combinatorial, variables $x_{k+1}, x_{k+2}, ..., x_n$ are continuous. The linear Euclidean problem of lexicographic combinatorial optimization on arrangements is to find pair $\langle L(x^*), x^* \rangle$ such that

$$L(x^*) = \underset{x \in R^n}{\text{lexmax}} \sum_{j=1}^{n} c_j x_j \text{ , } x^* = \arg \underset{x \in R^n}{\text{lexmax}} \sum_{j=1}^{n} c_j x_j \qquad (5)$$

under conditions (3)–(4), i.e. $L(x^*)$ is the maximum value of $L(x)$ as the variables range over the set (3), (4) and x^* is lexicographically larger than any other maximal.

3 The method of constructing lexicographic equivalence

One of the approaches to solving problem (3)–(5) is in the partition of a polyhedron into equivalence classes followed by directed search of the obtained classes . For linear completely combinatorial optimization problems on arrangements, this approach (method of constructing lexicographic equivalence) is justified in [12, 14]. The study [13] generalizes the equivalence relation, which is used to partition the polyhedral set into equivalence classes, and proposes algorithms for constructing lexicographic equivalence to solve mixed combinatorial linear problems.

Lexicographic equivalence of points of the space with respect to k-arrangements (the relation λ_k) is used as the equivalence relation in the method of constructing lexicographic equivalence. Elements of the quotient set with respect to the equivalence λ_k are called λ_k-classes. Each element of set $E^k_\eta(G)$ defines separate λ_k-class. Such classes are called combinatorial. Algorithms for constructing lexicographic equivalence involve directed search of λ_k-classes. Let us describe these algorithms (validation of these algorithms for solving mixed combinatorial problems is presented in [13]).

The first algorithm is used to solve a problem of finding a pair (5) under conditions (3), (4) and $L(x) \in A$ where A is a discrete set. This algorithm involves search of λ_k-classes whose representatives satisfy

$$\sum_{j=1}^{n} c_j x_j = \sigma^h \qquad (6)$$

where σ^h ($h = 0, 1, 2, ...$) are sequential elements of set A.

According to the second algorithm, the direct search of combinatorial λ_k-classes in lexicographically increasing order and lexicographically decreasing order is carried out. Classes whose representatives give the objective function a value smaller than result obtained at previous iterations, are excluded from the search.

The third algorithm is approximate and allows getting the objective function value that differs from the optimum by no more than a predetermined value. It involves search λ_k -classes whose representatives satisfy

$$\tau^h \le \sum_{j=1}^{n} c_j x_j \le \sigma^h \qquad (7)$$

where difference $\sigma^h - \tau^h$ ($h = 0,1,2,...$) decreases.

4 Description of the software product

The proposed algorithms of the method of constructing lexicographic equivalence for solving linear conditional problems of combinatorial optimization on arrangement (3) - (5) were implemented as software and experimentally investigated in solving the problem.

Software was developed in the Turbo Delphi environment. The program allows you to generate a series of problems with the given parameters, to solve problems by selected algorithms, to save the results of problem solving, etc.

Input data (multiset, coefficients of the objective function and additional constraints) are presented as text files. This allows you to access the same set of data more than once. Data is generated using standard Delphi language procedure.

Multiset was assigned by its base and primary specification. Elements of the base are generated in increasing order, maximum difference between adjacent elements is given. Multiplicities of elements of the base were specified less or equal to the number k of combinatorial variables because the choice of larger multiplicities does not affect the solving problem.

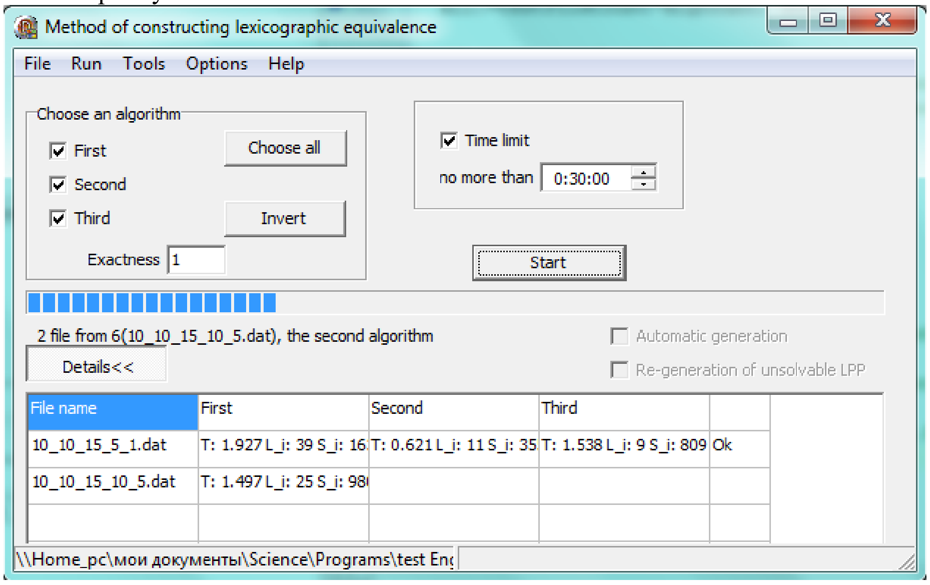
When creating data files, the following parameters were set:

- the number of elements of the multiset base;
- the maximum difference between adjacent elements;
- the dimension of space n ;
- the number k of combinatorial variables;
- the number m of constraints.

The main characteristics of the algorithms studied during computational experiments are the running time and the number of iterations in the process of approaching the solution. Time is defined as the difference between the system time before the start of the procedure that implements the corresponding algorithm and after its completion.

Before starting the test, the user must select the directory in which the problem files are placed. You can choose to test all the files contained in the selected directory, as well as individual files. You can also set the generation of files automatically before testing.

In the main window of the program (see Fig. 1), the user can select algorithms which will be used for solving problems. We assume that values of the objective function in the first algorithm of the method of constructing lexicographic equivalence should be integers (i. e. $A = Z$). If you choose the third algorithm you need to specify the exactness.


Method of constructing lexicographic equivalence

[✓ First]
[✓ Second]
[✓ Third]
Exactness 1

[✓ Time limit]
no more than 0:30:00

2 file from 6(10_10_15_10_5.dat), the second algorithm

Automatic generation
Re-generation of unsolvable LPP

File name	First	Second	Third	
10_10_15_5_1.dat	T: 1.927 L_i: 39 S_i: 16	T: 0.621 L_i: 11 S_i: 35	T: 1.538 L_i: 9 S_i: 809	Ok
10_10_15_10_5.dat	T: 1.497 L_i: 25 S_i: 98			

\\Home_pc\мои документы\Science\Programs\test En

Fig. 1. The main window of software tool for solving linear combinatorial optimization problems on arrangements.

Since even with rather small dimensions there are problems whose solving requires a considerable amount of time, the program allows you to set the time limit for the execution of algorithms. In case the algorithm runs longer than the specified time, the execution is interrupted and the corresponding message is output to the result file. If necessary, you can try to solve such problems later.

During testing, the main window shows the percentage of problems that are completed, the name of the current problem file and the name of the algorithm. When the "Details" button is clicked, a table of results is displayed. This table contains the names of the tested files, the time of solving the corresponding problem by each of the algorithms and the number of iterations.

In addition to solving existing problems by selected algorithms, the software product allows automatically generate series of problems with given parameters (the dimension of space, the numbers of constraints and combinatorial variables, the number of elements of the multiset base, etc.). The maximum and minimum values of each parameter of the series are specified, as well as the number of problems with each set of parameters.

All algorithms of the method of constructing lexicographic equivalence involve solving linear programming problem which is obtained by replacing the combinatorial condition (3) with the condition $(x_1, x_2, ..., x_k) \in \operatorname{conv} E_\eta^k(G)$ where $\operatorname{conv} E_\eta^k(G)$ is a convex hull of the set $E_\eta^k(G)$. If such a problem has no solution then the initial combinatorial optimization problem (3)–(5) also has no solution. In this case, the study of the effectiveness of the algorithms for constructing lexicographic equivalence does not make sense. Therefore the possibility of repeated generation is provided in the program. If a series received such problem that corresponding linear programming problem has no solution then file of this problem is generated again.

For the convenience of further analysis of the results of computational experiment, it is possible to create a summary file. This file contains for each problem information about the main parameters (the number of variables, the number of constraints, etc.) and the characteristics of the algorithms (the running time, the number of iterations). In the future, the data of such files can be systematized and grouped, for example, using a table processor.

5 Results of computational experiments

Several series of computational experiment were conducted using the software product described above. Testing of programs was carried out on a computer with a dual-core AMD A4-3400 processor, clock speed of each core 2700 MHz.

The first series of tests consists of 480 tasks, the generation of which parameters are given as follows: the dimension of space $n = 30$; number of combinatorial variables k accepts consecutive values from 15 to 30; number of constraints m accepts consecutive values from 5 to 10.

For each set of parameters, 5 problems were solved. Separate characteristics of the results of solving problems for which the running time does not exceed 30 min is given in the Table 1, Table 2, where such notation is used: qp is the percentage of problems solved in less than 30 minutes; st is the average running time; mt is the maximum running time.

Time is given in minutes:sec; time less than 1 second displayed as 0:01.

Table 1. The dependence of the running time of algorithms for constructing lexicographic equivalence on the number of constraints *m* (results of the first series of tests)

m	The first algorithm			The second algorithm			The third algorithm		
	qp	*st*	*mt*	*qp*	*st*	*mt*	*qp*	*st*	*mt*
5	100%	0:08	2:48	100%	0:40	12:58	100%	0:07	3:12
6	100%	0:11	4:41	100%	0:55	13:16	100%	0:10	4:16
7	100%	0:15	14:38	100%	0:51	15:33	100%	0:12	10:49
8	98,8%	0:33	10:56	97,5%	1:25	25:55	98,8%	0:33	14:27
9	97,5%	0:16	7:10	96,3%	1:04	16:09	97,5%	0:14	3:53
10	98,8%	0:43	9:28	93,8%	1:13	17:58	96,3%	0:25	7:07

Table 2. The dependence of the running time of algorithms for constructing lexicographic equivalence on the number of combinatorial variables *k* (results of the first series of tests)

k	The first algorithm			The second algorithm			The third algorithm		
	qp	*st*	*mt*	*qp*	*st*	*mt*	*qp*	*st*	*mt*
15	100%	0:01	0:06	100%	0:01	0:17	100%	0:01	0:04
16	100%	0:01	0:02	100%	0:02	0:13	100%	0:01	0:02
17	100%	0:01	0:12	100%	0:05	0:38	100%	0:01	0:10
18	100%	0:01	0:04	100%	0:03	0:22	100%	0:01	0:03
19	100%	0:01	0:06	100%	0:05	0:55	100%	0:01	0:06
20	100%	0:01	0:12	100%	0:05	1:38	100%	0:01	0:08
21	100%	0:01	0:10	100%	0:07	0:43	100%	0:01	0:10
22	100%	0:02	0:21	100%	0:19	4:09	100%	0:02	0:08
23	100%	0:02	0:27	100%	0:14	1:36	100%	0:02	0:12
24	100%	0:10	2:28	100%	1:03	12:25	100%	0:09	1:37
25	100%	0:11	2:44	93,3%	0:26	3:52	96,7%	0:05	0:44
26	100%	0:28	8:18	96,7%	1:44	9:36	93,3%	0:15	2:07
27	96,7%	0:32	9:28	96,7%	2:13	15:01	100%	0:29	7:07
28	96,7%	0:44	8:06	90%	2:04	12:58	96,7%	0:59	14:27
29	96,7%	1:24	7:36	100%	3:36	25:55	96,7%	0:49	3:53
30	96,7%	2:03	14:38	90%	4:43	17:58	96,7%	1:36	12:48

The results of the experiments presented in the Table 1, 2 show that the dependence of the running time of algorithms both on the number of combinatorial variables, and on the number of constraints, is characterized by a certain irregularity. In particular, all 30 problems with *k*=27 combinatorial variables were solved by the third algorithm in less than 10 minutes, whereas for two problems with *k*=26 the solution was not obtained in 30 minutes. The average running time of the second algorithm for problems with 22 combinatorial variables is greater than for problems with 23 combinatorial variables. Solutions 78 out of 80 problems with 9 constraints were obtained by the first algorithm in less than 30 minutes, while 79 out 80 problems with 10 constraints were solved in less than 30 minutes.

"Irregularity" confirmed Fig. 2, where each point corresponds to the results of the solving using the first algorithm for constructing lexicographic equivalence of one problem (the abscissa of the point is the number of combinatorial variables, and the ordinate is the corresponding time of solving the problem; if problem was not solved then time was equal to 30 min). For the second and third algorithms, similar images are obtained.

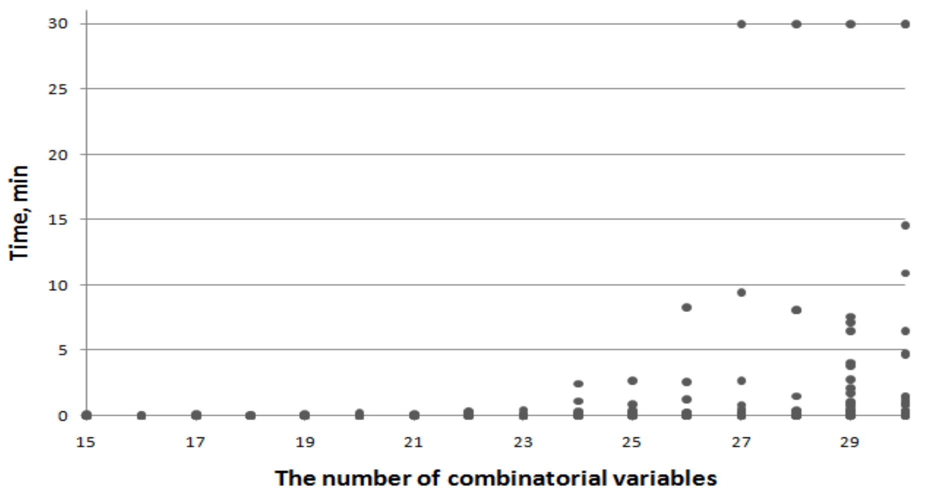


Fig. 2. The dependence of the running time of the first algorithm for constructing lexicographic equivalence on the number of combinatorial variables (results of the first series of tests)

The second series of tests consists of 420 problems (see Table 3).

Table 3. The dependence of the running time of algorithms for constructing lexicographic equivalence on the dimension *n* of space (results of the second series of tests)

n	The first algorithm		The second algorithm		The third algorithm	
	st	*mt*	*st*	*mt*	*st*	*mt*
30	0:11	2:16	0:43	8:57	0:11	2:56
31	0:16	4:49	0:53	10:46	0:11	3:26
32	0:01	0:08	0:13	1:54	0:01	0:09
33	0:02	0:20	0:19	4:04	0:02	0:17
34	0:10	4:37	0:34	4:43	0:04	1:00
35	0:02	0:14	0:12	2:03	0:02	0:15
36	0:02	0:21	0:18	1:53	0:02	0:12
37	0:01	0:04	0:17	4:44	0:01	0:06
38	0:01	0:13	0:20	3:19	0:02	0:17
39	0:01	0:05	0:09	1:28	0:01	0:14
40	0:01	0:09	0:18	3:26	0:01	0:13
41	0:01	0:06	0:15	1:59	0:01	0:09
42	0:01	0:23	0:08	2:00	0:01	0:16
43	0:01	0:02	0:05	1:03	0:01	0:03

The number of combinatorial variables is equal to 25, dimension accepts consecutive values from 30 to 43 and the number of constraints accepts consecutive values from 5 to 10, 5 problems were solved for each set of parameters. The running time was less than 11 minutes. The results are presented in the Table 3, where, as in the previous tables, *st* is the average running time; *mt* is the maximum running time. In contrast to the first series of tests if the dimension of space increases (so the number of continuous variables increases) then the running time decreases.

Parameters of problems in the third series of tests were defined as follows: the number of continuous variables is equal to 7, the number of constraints is equal to 10, dimension accepts consecutive values from 25 to 48 (so the number of combinatorial variables changes from 18 to 40), the number of problems with identical parameters is equal to 10.

The analysis of Table 3 and Table 4 gives reason to assert that the dimension of space has less effect on the running time than the number of continuous variables, but the corresponding dependencies with an acceptable correlation coefficient can not be established.

Table 4. The dependence of the running time of algorithms for constructing lexicographic equivalence on the dimension *n* of space (results of the third series of tests)

n	The first algorithm			The second algorithm			The third algorithm		
	qp	*st*	*mt*	*qp*	*st*	*mt*	*qp*	*st*	*mt*
25	100%	0:02	0:17	100%	0:04	0:12	100%	0:01	0:04
26	100%	0:08	0:32	100%	0:13	0:37	100%	0:04	0:13
27	100%	0:02	0:08	100%	0:12	1:18	100%	0:01	0:04
28	100%	0:02	0:05	100%	0:08	0:31	100%	0:01	0:03
29	100%	0:13	1:51	100%	0:14	0:44	100%	0:04	0:21
30	100%	0:13	1:09	100%	0:55	3:27	100%	0:12	0:48
31	100%	0:07	0:22	100%	0:51	3:22	100%	0:06	0:17
32	100%	0:21	1:46	100%	1:06	2:19	100%	0:11	0:31
33	100%	0:08	0:26	90%	0:22	1:25	100%	0:08	0:41
34	100%	0:13	1:04	100%	1:52	6:58	100%	0:24	2:32
35	100%	0:59	7:02	100%	1:36	6:35	100%	0:31	2:60
36	100%	0:06	0:31	100%	2:49	12:04	100%	0:11	0:33
37	100%	0:03	0:11	100%	0:49	2:49	100%	0:04	0:16
38	100%	0:18	2:12	90%	1:36	7:20	90%	0:09	0:29
39	100%	0:10	0:32	80%	1:54	7:37	90%	0:27	2:27
40	100%	0:04	0:19	90%	0:29	2:08	100%	0:11	1:03
41	90%	0:14	0:59	90%	1:57	12:42	100%	2:57	25:37
42	100%	0:27	1:44	90%	7:19	26:38	100%	0:40	2:34
43	100%	1:26	4:48	70%	3:53	18:48	100%	1:36	6:52
44	80%	0:01	0:04	80%	2:27	8:17	80%	0:09	0:25
45	100%	0:58	4:41	90%	4:33	22:52	100%	1:14	5:25
46	100%	0:11	1:01	90%	2:37	12:53	100%	0:22	1:17
47	100%	0:11	0:28	90%	4:23	14:42	100%	0:28	1:53
48	90%	0:21	0:42	60%	12:01	28:54	90%	1:25	3:43

6 Conclusion

The paper considers the software implementation of the algorithms of the method of constructing lexicographic equivalence. This method is used for solving linear mixed combinatorial optimization problems on arrangements and involves partition the space into equivalence classes followed by their direct search. The offered software allows you to generate a series of problems with the given parameters, to solve problems by selected algorithms, to save the results of problem solving. The computational experiment showed that the developed algorithms are effective for most problems with dimensions up to 50.

References

1. Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: Algorithms and Complexity. Dover Publications, Mineola, New York (1982)
2. Grötschel, M., Lovász, L., Schrijver, A.: Geometric algorithms and combinatorial optimization. Springer-Verlag, Berlin Heidelberg (1993)
3. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms. Algorithms and Combinatorics, vol.21. Springer, Berlin, Heidelberg (2018) doi: 10.1007/978-3-662-56039-6
4. Nemhauser, G. L., Wolsey, L. A.: Integer and combinatorial optimization. John Wiley & Sons, New York (1988)
5. Pardalos, P.M., Du, D.-Z., Graham R.L. (eds.): Handbook of Combinatorial Optimization, Springer-Verlag, New York (2013)
6. Hulianytskyi, L., Riasna, I.: Formalization and Classification of Combinatorial Optimization Problems. In: Butenko S., Pardalos P., Shylo V. (eds): Optimization Methods and Applications. Springer Optimization and Its Applications, vol 130, pp. 239-250. Springer, Cham (2017) doi: 10.1007/978-3-319-68640-0_11
7. Zgurovsky, M.Z., Pavlov. A.A.: Combinatorial Optimization Problems in Planning and Decision Making. Studies in Systems, Decision and Control, vol. 173. Springer, Cham (2019) doi: 10.1007/978-3-319-98977-8
8. Emets', O. O., Roskladka, O. V., Nedobachii S. I.: Irreducible System of Constraints for a General Polyhedron of Arrangements. Ukr. Mat. Zh.. 55(1): 1-12 (2003) doi: 10.1023/A:1025060316418
9. Stoyan, Yu. G., Iemets, O. O.: Theory and methods of Euclidean combinatorial optimization [in Ukrainian]. Instytut systemnykh doslidzhen osvity, Kyiv (1993). http://dspace.puet.edu.ua/handle/123456789/487
10. Semenova, N. V., Kolechkina, L. N., Nagornaya, A. N.: One Approach to Solving Vector Problems with Fractionally Linear Functions of the Criteria on the Combinatorial Set of Arrangements. J. Automat. Inform. Sci. 42(2): 67-80 (2010) doi: 10.1615/JAutomatInfScien.v42.i2.50
11. Yakovlev, S. V., Grebennik, I. V.: Some classes of optimization problems on a set of arrangements and their properties. Izvestiya Vysshikh Uchebnykh Zavedenii. Matematika 11: 74-86 (1991)
12. Iemets, O. O., Barbolina, T. M.: Combinatorial optimization on arrangements [in Russian]. Naukova dumka, Kyiv (2008) http://dspace.puet.edu.ua/handle/123456789/473
13. Barbolina, T. N. Solution of mixed combinatorial optimization problems on arrangements by the method of construction of lexicographic equivalence. Cybern. Syst. Analysis 49 (6): 137-149 (2013) doi: 10.1007/s10559-013-9582-4
14. Yemets, O. A., Barbolina, T. N.: Solution of euclidean combinatorial optimization problems by the method of construction of a lexicographic equivalence. Cybern. Syst. Analysis 40 (5): 76-734 (2004) doi: 10.1007/s10559-005-0010-2