ECONTECHMOD. AN INTERNATIONAL QUARTERLY JOURNAL – 2019, Vol. 08, No. 1, 33-38

Synthesis of Recovery Schemes for Distributed Computing Based on Ideal Ring Bundles

Riznyk O[1]., Vynnychuk R.[2]

[1]Department of Publishing Information Technologies, Lviv Polytechnic National University, S. Bandery 28a, 79008 Lviv, Ukraine; e-mail: riznykoleg@gmail.com

[2]Department of Human Resource Management and Administration, Lviv Polytechnic National University, S. Bandery 28a, 79008 Lviv, Ukraine; e-mail: vynnychuk.roksolana@gmail.com

Received February 08.2019: accepted March 11.2019

Abstract. Clusters and distributed systems allow for error tolerance and high performance through shared use of the load. When all computers work, we would like to distribute the load equally among computers. When one or more computers are broken, the load on these computers should be redistributed to other computers in the cluster. Redistribution determines the recovery scheme. The recovery plan should keep the load as optimal as when even the most unfavorable combinations of computers are turned off, that is, we want to optimize the behavior of the worst case. In this work, we find new regeneration schemes based on so-called IRB and ideal ring loops and synthesize the corresponding combinatorial structures. They are optimal for many cases of recovery schemes.

Keywords: cluster, distributed system, Golomb ruler, ideal ring bundle, recovery scheme.

INTRODUCTION

Application of recovery schemes in distributed computing for information systems is due to the complexity of such modern systems and the need to prevent errors in working in a distributed network when parts of computers are abandoned [4, 12].

Load balancing and suitability are important in distributing online trusted systems. Krishna and Shin define tolerance for error as "the ability of the system to respond immediately to an unexpected error of the technical support or program" [13]. Many compromise-tolerant systems reflect all actions, for example, each action is executed on two or more dual systems in such a way that, if one does, the other can accept it. The advantage of using clusters, apart from tolerance to error, is the load that is shared between computers [3, 14].

One can manage this task dynamically, where transfer decisions do not depend on the actual state of the system. The task with dynamic politics is rather unpredictable.Otherwise - you should use a static policy, which is generally based on the information behavior of the system. Here, the transfer solution is independent of the actual current state of the system. This makes them less complex and more predictable than dynamic policies [2, 4].

The task, studied in this work, is: how to evenly distribute the current computers when one or more computers in the cluster are turned off and how to build these mathematical models most importantly? Let's consider it a static redistribution of work.

FORMULATION OF THE PROBLEM

There is a main computer that executes the application under normal conditions and a secondary computer that accepts the task when the host computer is turned off. Perhaps there is also a third computer that takes a task when the main and secondary computers are off, and so on. When all computers are working, we take steps to spread the load evenly [3]. The load on some computers will, however, grow when one or more computers are shut down, but under these conditions we would like to distribute the load as evenly as possible on the remaining computers.

The distribution of the load when the computer is off is resolved by a list of recovery processes that are performed on an erroneous computer. The set of all recovery lists is called recovery. Consequently, the distribution of the load is completely determined by the recovery scheme regardless of the number of computers that are turned off [5, 12].

The problem of identifying optimal (or even better) recovery schemes is studied in [4, 6]. We will find recovery schemes that guarantee the load balances for more computers in the cluster. This work looks at, including the worst case scenario, when a computer that is already redistributing another process from a pre-broken computer has failed.

SOLVING THE PROBLEM KEY

We assume that the work performed by each of the N computers must be moved as one atomic unit. This is a very common scenario, which includes cases where:

• The work performed by the computer is done from one external system (stacked recovery schemes work in external systems, such as the second, third, fourth, etc., alternative locations (alternative cluster nodes)

destination, if the main, secondary, etc. the destination computer is disabled);

• There is one network address for each computer and we use the IP protocol (or similar technique);

• All work done by the computer, does one process or group of related processes using shared internal resources and therefore must be moved as one unit.

In [14], a case is considered when a number of independent processes are performed by each computer. These results can be easily generalized to the case with the number of independent processes on each computer using the same technique.

Consider a cluster with n identical nodes, which under normal conditions perform one process each. All processes perform the same amount of work. We see a schema recovery for the cluster that determines where the process should start again if its current node is off. The recovery scheme should keep the load balanced to the cluster when even the most unfavorable combination of nodes is off. The recovery scheme must be calculated for a large n in real time.

The recovery scheme R consists of n restores lists - one list per process. Let R be a sequence of distances, that is, a sequence of positive integers $S =< s_1,\ s_2,\ ...,\ s_{n-1} >$ where $\sum_{j=1}^{i} s_j \ (i = 1, \ldots, n-1)$ - the distance between the damaged node and the node at which process must be restarted in the i-th step. Let R_0 be a process restore 0 (which runs on a computer 0). The list of restorations from which we construct the sequence S:

$$R_0 =< 0,\ s_1,\ s_1 + s_2,\ s_1 + s_2 + s_3,\ ...,\ s_1 + ... + s_{n-1} > \quad (1).$$

When one node in the cluster breaks down, the load on the most loaded node in the cluster (we call that node Z) will contain two jobs: in fact, it's work from the node. When two nodes in the cluster are turned off, there are two possibilities. If both are initially in sequence, the load on Z can contain three jobs: its own, from one node and from another node [3, 5]. This situation is presented in Fig. 1a. Nodes and - damaged nodes. The second option is - when the first two numbers in the sequence are different. In this case, Z can contain only two works: its own and from one node or from another node. This situation is presented in Fig. 1b. Therefore, the load in the worst case on Z is less than when $s_1 \neq s_2$. Fig. 1b also shows the situation when three nodes in the cluster are turned off.

Then the load on Z can contain three jobs: its own and from nodes. Note that in this case the node is also a damaged node. The worst case for the three damaged nodes is when the first of the three numbers in the sequence is equal, that is, $s_1 = s_2 = s_3$. In this case, Z will contain four works: its own and from the knots.

In [6], authors consider the task of detecting a recovery scheme that can guarantee optimal load distribution in the worst case when most k computers are turned off. Schemes should have as much as possible a k. Authors present and prove it in the Log algorithm, it does a recovery scheme that guarantees optimality, where the largest number of $log_2 n$ computers is disabled. The optimal means when the maximum number of processes on the same computer after k offsets is $MV(k)$, where $MV(k)$ provides the lower limit for any static recovery scheme [4, 6].

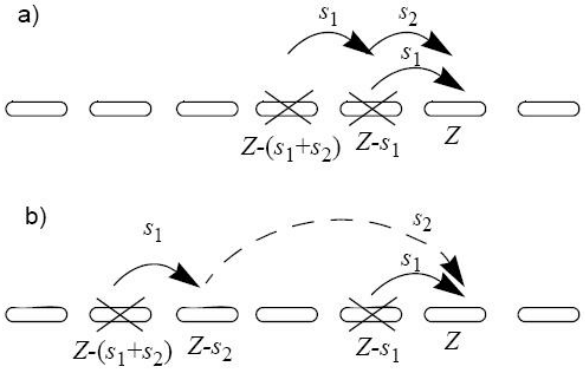

Fig. 1. The script with three broken knots.

Another algorithm, called saving, is presented in [10]. This algorithm generates recovery schemes that give optimality to a greater number of cases than the Log algorithm (that is, saving provides optimal guarantees when more than 2 logins are off.) The saving algorithm is based on the mathematical problem of identifying a sequence of positive integers such that the entire sum of the sequences is unique and minimal.

It`s simple for the case when it is necessary to calculate a saving algorithm, even for a large n. The list of restorations for a zero process, running on a computer with zero, consists of two parts. The first part is a sequence of thrifty algorithms, and the second part is filled with the remaining numbers. The recovery scheme consists of all recovery lists. A separate case of the Saving algorithm is the algorithm called the ideal ring bundle (IRB), described in [15]. The name of this algorithm comes from the IRB, which is a sequence of positive integers such that there are no two identical numbers or adjacent sums of them. These numbers are labeled and correspond to computer positions. The difference between the values of any two labels is called distance. The shortest construction for this series of labels is called the ideal ring bundle [12]. The search for the IRB becomes more difficult as the number of tags increases. This is known as an NP-complete task [8, 16]. The task of detecting IRBs for a large number of labels is still unresolved.

A simple IRB is called the sequence $L_N = (l_1,\ l_2,\ ...,\ l_N)$ of numbers on which all possible ring amounts exhausts the values of numbers of the natural series from 1 to $S_N = N(N-1)+1$.

An example of a six-dot IRB is shown in Table I. It is possible to measure all $S_6 = 31$ distances, that is, it is suitable for synthesis of the recovery scheme with $S_6 + 1 = 32$ computers (32, because we take into account

that the countdown starts with a zero computer), so $n = S_N + 1$.

TABLE I. Row amounts for IRB in the sixth order (1, 3, 2, 7, 8, 10)

	p_j q_j					
	1	2	3	4	5	6
1	1	4	6	13	21	31
2	31	3	5	12	20	30
3	28	31	2	9	17	27
4	26	29	31	7	15	25
5	19	22	24	31	8	18
6	11	14	16	23	31	10

The sequence (1, 3, 2 7, 8, 10) is one of the simplest perfect rings of the 6th order. IRBs can also be represented as geometric shapes, where each number represents the sum between pairs of nearby rooms.

In IRB, cyclic transitions are ignored, that is, situations when the result of the number of "jumps" for the process is greater than the number of computers in the cluster. "Jumping" is the distance between the damaged node and the node in which the process needs to be restarted. Including cyclic transitions give a new mathematical formulation with the detection of the longest sequence of positive integers such that the sum and sum of the entire module of sequences (including sequences of length one) n are unique. This mathematical formulation of the computer problem poses a new, more powerful, IRB-based recovery scheme for a larger number of corrupted computers than known modular schemes.

All these algorithms (Log, Savings, IRB and Modular) guarantee the optimality of a certain number of corrupted computers in the cluster. [3, 5] discusses the best possible recovery schemes for any number of corrupted computers. Because of the complexity of the problem, finding such a recovery scheme is a complex task. We only have the optimal recovery scheme for a maximum of 21 computers in the cluster $(n \le 21)$ [6].

New regeneration schemes based on optimal IRB lines guarantee optimal behavior for a much larger number of corrupted nodes [12].

Scheme of recovery IRB - a regular recovery scheme. For n nodes in a cluster, we build a list of restorations using a known IRB with a sum of less than or equal to S_N, and the rest of the recovery list is filled with the remaining numbers up to $S_N + 1$, for example, for $n = 13$ we have a list < 0, 1, 3, 9, 13, 2, 4, 5, 6, 7, 8, 10, 11, 12>.

Recovery schemes are based on IRB rulers and are filled with numbers that construct new collapsing routes. IRB recovery schemes provide better optimization than circuits on the Golombs rulers [1, 7].

The algorithm for constructing ideal ring bundles with the accompanying matrix A of a field $GF(p^S)$ of a polynomial $f(x)$ in a natural basis involves the use of a matrix method for determining the coordinates of elements of the field $GF(p^S)$, in which each S-dimensional vector-column of coordinates elements of the field $GF(p^S)$ are found as the result of multiplying the matrix A into another vector-column.

The accompanying matrix of a polynomial $f(x) = x^s + a_{s-1}x^{s-1} + \ldots + a_1 x + a_0$ has the following form:

$$A = \begin{Vmatrix} 0 & 0 & \ldots & 0 & a_0 \\ 1 & 0 & \ldots & 0 & a_1 \\ 0 & 1 & \ldots & 0 & a_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 & a_{S-2} \\ 0 & 0 & \ldots & 1 & a_{S-1} \end{Vmatrix}. \quad (2)$$

The algorithm of synthesis consists of the following sequence of operations:

1. Find the original polynomial *f(x)* of the third degree, which is irreducible over the field $GF(p^\alpha)$, where *p* is a simple number determined from the equation $N = p^\alpha + 1$, where n is the order of the IRB, and *S* is an integer.

2. Write the accompanying matrix A of the original polynomial $f(x) = x^3 + a_2 x^2 + a_1 x + a_0$ irreducible over a field $GF(p)$ in the natural basis:

$$A = \begin{Vmatrix} 0 & 0 & -a_0 \\ 1 & 0 & -a_1 \\ 0 & 1 & -a_2 \end{Vmatrix}. \quad (3)$$

3. Multiplying the matrix A by the vector column $b_1 = \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix}$, write the following vector-column b_2. Continue to fill $S_n = N(N-1)+1$ by successive columns, where each subsequent column is the result of multiplying the matrix A of the previous column.

4. Write the sequence numbers $i_1, i_2, \ldots, i_n$ in vector columns having zero in the last line, and find the elements of the ring link according to the formula:

$$k_i = \begin{cases} b_{i+1} - b_i, if & i = \overline{1, N-1} \\ b_1 - b_n \pmod{S_n}, \ if\ i = N \end{cases}. \quad (4)$$

5. Check whether the ring ring found is perfect. To do this we build a special table, which, in the case of IRB, each number in the range (1, 2, ..., S) is R times.

To construct IRB by this algorithm, primitives are required for the field $GF(p^S)$ polynomials of degree 3 that are irreducible. It should be noted that for different fields, different methods of constructing polynomials are used, which greatly complicates the implementation of algorithms for the synthesis of perfect knots based on the

use of the properties of expanded fields of Galois.

Another problem that faces when generating perfect families of perfect knots based on this technique is to split the set of coefficients of the corresponding difference set into non-overlapping classes of coefficients, which are used as factors to find all the variants of ideal knots without exception [10, 12]. However, for the general case this task is not solved. Therefore, using classical methods, including group theoretical and theoretical-numerical, intensive search for fundamentally new approaches to solving the issue of generating complete families of perfect relationships [9, 11].

RESEARCH RESULTS

In a number of clustered and distributed systems the projector has to come up with a recovery system, which defines, what operating capacity has to be redistributed in case of one or a few computers are shut down. The objective is to get a capacity, which would equally redistribute under any scenarios, even when the most unfavorable combinations of computer failures are reached. We are studying a cluster with identical n nodes, which under usual circumstances do one process per node. All processes conduct an equal amount of operations. The recovery scheme guarantees the most favorable capacity distribution in the worst scenario, when k computers are taking damage. Recovery schemes which are presented here can also be used in clusters with a number of independent processes on each computer. New recovery schemes are offered, where the first part of scheme is based on well-known IRB (collapsing routes do not overlap), and the other part has a projected path, where the next collapsing routes are unique in relation to the previous collapsing routes.

The IRB recovery schemes guarantee better implementation than other existing schemes and are simple in calculation. There is no difference between these schemes if there are few nods in the cluster. But for a bigger cluster the IRB recovery scheme's behavior is better than other existing IRB schemes. For instance, for $n = S_{20} + 1 = 381 + 1 = 382$ the IRB scheme guarantees an optimal worst behavior case for $k = N + 1 = 20 + 1 = 21$ shutdowns. The objective of the work is to find appropriate IRB based recovery schemes, which would be better than the already existing ones and which could easily calculate a recovery for a hign n of nods in a cluster on Golomb rulers. Indicating that such a recovery scheme is equally complicated in terms of calculation synthesis is a difficult task [1, 2].

The IRB recovery schemes can already be used in the commercial cluster system, defining a list in SunCluster, via using the *scconf* command. The results can also be used in external systems like telecommunication centers, which switch and send data to different nods in a distributed system (or clusters, where nods possess individual network addresses) [4, 14, 15, 17, 18]. With the help of quick IRB synthesis analysis, based on Galois field machine.

The first parts of recovery schemes are built on the basis of ideal ring bundles, which are noted in Table II.

The second part of the recovery system based on ideal ring bundles is built for an IRB with a higher number of elements (computers of a divided system), which are noted in Table III.

TABLE II. FIRST PARTS OF IRB-BASED RECOVERY SCHEMES ($N < 21$)

Amount of sequences S_N	N	First part of recovery schemes on the basis of ideal ring bundles
13	4	0, 1, 3, 9, 13
21	5	0, 1, 4, 14, 16, 21
31	6	0, 1, 3, 8, 12, 18, 31
57	8	0, 1, 3, 13, 32, 36, 43, 52, 57
73	9	0, 1, 3, 7, 15, 31, 36, 54, 63, 73
91	10	0, 1, 3, 9, 27, 49, 56, 61, 77, 81, 91
133	12	0, 1, 3, 12, 20, 34, 38, 81, 88, 94, 104, 109,
183	14	0, 1, 3, 16, 23, 28, 42, 76, 82, 86, 119, 137, 154, 175, 183
273	17	0, 1, 3, 7, 15, 31, 63, 90, 116, 127, 136, 181, 194, 204, 233, 238, 255, 273
307	18	0, 1, 3, 21, 25, 31, 68, 77, 91, 170, 177, 185, 196, 212, 225, 257, 269, 274, 307
381	20	0, 1, 3, 12, 17, 65, 75, 94, 117, 124, 132, 145, 163, 167, 200, 271, 297, 303, 337, 357, 381

The developed software that simulates the work of the recovery scheme based on IRB with the redistribution of processes. After running the program before starting to synthesize recovery schemes, you must enter:

1. In the application, the entry parameters of an IRB of N size (scales indicators of N elements of an IRB).

2. In the application, the number of computers, which are used in distributed computing, with amount to the sum of IRB elements S_N plus one.

3. To enter the computer's number, which has taken damage and does not operate.

CONCLUSIONS

In this work, the complex of identical computers was studied, which, under normal circumstances, conduct one process. All the processes conduct the same amount of operations. It was researched that the task of finding the most favorable recovery schemes with n computers matches the mathematical task of finding the longest sequence of positive integers, where all the numbers ale located and are unique and match an ideal ring bundle. In this work, such recovery schemes are studied, which optimally fit for the majority of turned off computers compared to existing schemes like those based on Golomb ruler. A program product is established and tested for synthesis of recovery scheme based on ideal ring bundles.

TABLE III. SECOND PARTS OF IRB-BASED RECOVERY SCHEMES ($N < 21$)

Amount of sequences S_N	N	Second part of recovery schemes on the basis of ideal ring bundles
13	4	2, 4 – 8, 10 12
21	5	2, 3, 5 – 13, 15, 17 - 20
31	6	2, 4 – 7, 9 – 11, 13 – 17, 19 - 30
57	8	2, 4 – 12, 14 – 31, 33 -35, 37 – 42, 44 – 51, 53 -56
73	9	2, 4 – 6, 8 – 14, 16 – 30, 32 – 35, 37 – 53, 55 – 62, 64 - 72
91	10	2, 4 – 8, 10 – 26, 28 – 48, 50 – 55, 57 – 60, 62 – 76, 78 – 80, 82 - 90
133	12	2, 4 – 11, 13 – 19, 21 – 33, 35 – 37, 39 – 80, 82 – 87, 89 – 93, 95 – 103, 105 – 108, 110 - 132
183	14	2, 4 – 15, 17 – 22, 24 – 27, 29 – 41, 43 – 75, 77 – 81, 83 – 85, 87 – 118, 120 – 136, 138 – 153, 155 – 174, 176 - 182
273	17	2, 4 – 6, 8 – 14, 16 – 30, 32 – 62, 64 – 89, 91 – 115, 117 – 126, 128 – 135, 137 – 180, 182 – 193, 195 – 203, 205 – 232, 234 – 237, 239 – 254, 256 - 272
307	18	2, 4 – 20, 22 – 24, 26 – 30, 32 – 67, 69 – 76, 78 – 90, 92 – 169, 171 – 176, 178 – 184, 186 – 195, 197 – 211, 213 – 224, 226 – 256, 258 – 268, 270 – 273, 275 - 306
381	20	2, 4 – 11, 13 – 16, 18 – 64, 66 – 74, 76 – 93, 95 – 116, 118 – 123, 125 – 131, 133 – 144, 146 – 162, 164 – 166, 168 – 199, 201 – 270, 272 – 296, 298 – 302, 304 – 336, 338 – 356, 358 - 380

REFERENCES

1. **Brglez F., Bošković B. and Brest J.**, **2017**. On asymptotic complexity of the optimum Golomb ruler problem: From established stochastic methods to self-avoiding walks, IEEE Congress on Evolutionary Computation (CEC), San Sebastian. 1000-1007. doi: 10.1109/CEC.2017.7969417.
2. **Vyas J., Bansal S. and Sharma K.**. **2016**. Generation of optimal Golomb rulers for FWM crosstalk reduction: BB-BC and FA approaches," 2016 International Conference on Signal Processing and Communication (ICSC), Noida, 74-78. doi: 10.1109/ICSPCom.2016.7980551.
3. **Klonowska, K., Lundberg, L., Lennerstad, H., Svahnberg, C.**: **2004**. Using Modulo Rulers for Optimal Recovery Schemes in Distributed Computing, in Proceedings of 10th International Symposium PRDC Papeete, Tahiti, French Polynesia, March. 133-142.
4. **Lundberg, L., Häggander, D., Klonowska, K., Svahnberg, C.**, **2003** Recovery Schemes for High Availability and High Performance Distributed Real-Time Computing, in Proceedings of 17th International Parallel & Distributed Processing Symposium IPDPS 2003, Nice, France, April 2003, pp. 122.
5. **Lundberg, L., Häggander, D., Klonowska, K., Svahnberg, C.**, **2003**. Recovery Schemes for High Availability and High Performance Distributed Real-Time Computing, in Proceedings of 17th International Parallel & Distributed Processing Symposium IPDPS 2003, Nice, France, April 2003, pp. 122, CD-ROM.
6. **Lundberg, L., Svahnberg, C. 2011**. Optimal Recovery Schemes for High-Availability Cluster and Distributed Computing, Journal of Parallel and Distributed Computing, 61(11), 2011, pp. 1680-1691.
7. **Memarsadeghi N. 2016.** NASA Computational Case Study: Golomb Rulers and Their Applications," in Computing in Science & Engineering, vol. 18, no. 6, pp. 58-62, Nov.-Dec. 2016. doi: 10.1109/MCSE.2016.118.
8. **Oshiga O., Severi S. and Abreu G. T. F. de**. **2016**. Superresolution Multipoint Ranging With Optimized Sampling via Orthogonally Designed Golomb Rulers. In IEEE Transactions on Wireless Communications, vol. 15, no. 1, pp. 267-282, Jan. 2016. doi: 10.1109/TWC.2015.2470687.
9. **Riznyk O., Balych B. and Yurchak I. 2017**. A synthesis of barker sequences is by means of numerical bundles," 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Lviv, 2017, 82-84. doi: 10.1109/CADSM.2017.7916090.
10. **Riznyk O., Povshuk O., Kynash Y. and Yurchak I. 2017.** Composing method of anti-interference codes based on non-equidistant structures. 2017 XIIIth International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH), Lviv, 2017, pp. 15-17. doi: 10.1109/MEMSTECH.2017.7937522.
11. **Riznyk O., Parubchak V. and Skybajlo-Leskiv D.**, "Information Encoding Method of Combinatorial Configuration," 2007 9th International Conference - The Experience of Designing and Applications of CAD Systems in Microelectronics, Lviv-Polyana, 2007, pp. 370-370. doi: 10.1109/CADSM.2007.4297583.
12. **Riznyk O., Kynash Y., Povshuk O. and Kovalyk V. 2016**. Recovery schemes for distributed computing based on BIB-schemes. In 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv, 2016, 134-137. doi: 10.1109/DSMP.2016.7583524.
13. **Kaczmarek P., Mańkowski T. and Tomczyński J.**, **2017**. Towards sensor position-invariant hand gesture recognition using a mechanomyographic interface," 2017 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, 2017, 53-58. doi: 10.23919/SPA.2017.8166837.
14. **Li P., Dong J., Liu X., Wang G., Li Z. and Liu X.**. **2017**. PITR: An Efficient Single-Failure Recovery Scheme for PIT-Coded Cloud Storage Systems," 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), Hong Kong, , pp. 259-261. doi: 10.1109/SRDS.2017.38.
15. **R. Oleg, K. Yurii, P. Oleksandr and B. Bohdan**. **2017**. Information technologies of optimization of structures of the systems are on the basis of combinatorics methods," 2017 12th International

Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, 232-235. doi: 10.1109/STC-CSIT.2017.8098776.

16. **Blackburn S. R. and Etzion T.. 2017**. "PIR array codes with optimal PIR rates," 2017 IEEE International Symposium on Information Theory (ISIT), Aachen, 2017, 2658-2662. doi: 10.1109/ISIT.2017.8007011.
17. **Shakhovska N., Veres O., Hirnyak M. 2016.** Generalized formal model of big data. ECONTECHMOD 5, (2), 33–38.
18. **Bobalo Y., Politanskyi R., Klymash M. 2015.** Traffic simulation in a telecommunication system based on queuing systems with different input flows. ECONTECHMOD 4, (1), 11–15.