Large Discrete Systems Evolutionary Synthesis Procedure

David Aregovich Petrosov[1], Vadim Alexandrovich Lomazov[1], Sergey Igorevich Matorin[2], Alina Ivanovna Dobrunova[1] and Valentina Ivanovna Lomazova[2]

[1]Belgorod State Agricultural University Named After V. Gorin,
Russia, 308503, Belgorod region, pos. Mayskiy, ul. Vavilova 1
[2]Belgorod National Research University, Russia, 308015, Belgorod, ul. Pobedy, 85

DOI: http://dx.doi.org/10.13005/bbra/1841

(Received: 06 March 2015; accepted: 18 April 2015)

The article discusses the problems of structural synthesis of large discrete systems with predefined behavior, which assumes transition of a given input signal into required reference output signal. A combined method for building the procedure of synthesis based on evolutionary methods and mathematical analysis of Petri nets has been proposed. An evolutionary procedure of structural synthesis of large discrete systems with static inter-component links has been developed. Computational experiments performed with the use of the built model give evidence to the efficiency of the proposed synthesis procedure.

Key words: System analysis, structural synthesis of large discrete systems, evolutionary algorithms, genetic algorithm, Petri nets, simulation modeling.

Structural-parametric synthesis of large discreet systems with specified behavior is a variation of the more generic scientific and practical problem: synthesis of systems with predefined behavior that are able to transform a given input vector (data, substance, energy, etc.) into the required output vector[1-3].

This paper will propose an approach for solving the task of structural synthesis of large discrete systems with predefined behavior, which is based on three major theories: imitation modeling, evolutionary methods and Petri nets. Choice of these tools for solving the task of structural and parametric synthesis is explained by the fact that the majority of tasks in this area may be related to the multi-criteria class, since in solving them it is necessary to consider many factors. In solving tasks of this kind, one should pay attention not only to the structure of large discrete systems (sets of elements in the links between them) but to adjustment of elements in the possible range of their functioning, as well[1]. The essence of this problem lies in the fact that in the presence of 10 components and about 10 instances of each component, the number of possible configurations would be 10^10 (without taking into account the possibility of reconfiguration by dynamic changing elemental links, or changing parameters of elements operation). This significantly complicates searching for the required configuration of a large discrete system with predefined behavior in this area (in case of a random search, configuration becomes virtually impossible to find), therefore the method of introducing elements of determination into random search was adopted. For this purpose, the most suitable are evolutionary methods[4-6].

The most common methods for this class of tasks are genetic algorithms and genetic programming (using these methods makes it

* To whom all correspondence should be addressed.

possible to reduce the time of searching for a large discrete system configuration that works not only with changing elements of the system, but with the links between them that change during operation), and changes in elements settings. These methods require adaptation to the subject area. This paper proposes to use the theory of nested Petri nets (which are a rather new mathematical tool) as a tool for adapting the genetic algorithm. This will make it possible to combine two approaches: genetic programming and genetic algorithms[4] due to the use of the following properties: nesting (describing genetic algorithm at top level and presentation of labels of subsequent levels as genotypes, i.e., models of synthesized large discrete systems with predefined behavior), and building trees of achievable labels (tree-like coding in genetic programming).

Methodology

Genetic algorithms

Genetic algorithms reflect the principles of natural selection and genetics: survival of the most promising individuals, inheritance, and mutations. With that, there is a possibility to influence the search process by setting certain parameters[3-6].

The main differences between genetic algorithms and traditional methods are as follows:

1. Genetic algorithms deal with solutions presented as a line of code. Codes are converted without regard to their semantics[4-11].
2. The search process is based on using several points in the space of solutions simultaneously. This eliminates the possibility for an efficiency function, which is not unimodal, from unwanted getting into the local extremum.
3. In searching, the genetic algorithm uses only the information about permissible values of parameters and the efficiency function, which leads to a considerable increase in speed of response.
4. For the synthesis of new points, the genetic algorithm uses probability rules, and for transfer from points to points - deterministic rules. Such a combination of rules is much more efficient, as compared to using them separately.

The theory of the genetic algorithms also uses a number of biological terms.

A code string describing the possible solution, and its structure, is called a genotype. Code interpretation from the point of view of the solved task is called a phenotype. For the subject area in question, a phenotype would be some design solution in the form of a structural diagram of a computing device. The code is also called a chromosome.

A set of chromosomes used simultaneously by the genetic algorithm at each stage of the search is called a population. The size of population (number of chromosomes) is usually recorded and used as one of genetic algorithm characteristics. A population is updated by creating new chromosomes and deleting old ones. This is how generations of populations change.

Generating new chromosomes is based on modeling the reproduction process: a pair of parents produces a pair of children. The crossover operator is responsible for generation, and in general, it is used for every pair of parents with a certain probability. The value of such probability, along with the size of the population, is one of the characteristics of a genetic algorithm.

The mutation operator is applied to the chromosomes of the new population. The probability of applying this operator to the chromosome is also a parameter of the genetic algorithm.

The selection operator performs selection of parents' crossovers for producing children, and the reproduction operator is responsible for selecting chromosomes to be deleted. In both cases, selection is made based on chromosome quality, which is determined by the value of efficiency function of this chromosome.

The genetic algorithm stops working in the following cases:

1. The number of generations that the user has defined before starting the algorithm has been processed.
2. Quality of all chromosomes exceeded the value defined by the user before starting the algorithm.
3. The chromosomes have become

homogeneous to the extent that their improvement from generation to generation occurs very slowly.

Work of the genetic algorithm in terms of the subject area of structural synthesis of large discrete systems in question may be described as follows:

The initial set of configurations is formed by the experts. These solutions are subsequently coded into chromosomes that will be directly used by the genetic algorithm. It is in this stage already that the actual task of selecting the way of presenting the structural diagram of the synthesized system in form of a code line processed by the genetic algorithm occurs.

The parent chromosomes for reproduction are selected, and chromosomes deleted in accordance with the natural "The Strongest Survive" principle. At this stage, the problem of setting selection and reduction by the operator is important, which reflect peculiarities of systems structural synthesis. A similar problem occurs in defining the crossover operator, which should model the transfer of properties between various structural schemes of the system.

The genetic algorithms often use the elitism strategy that deals with transferring the best chromosomes of the current population to the next population without changes. Such an approach ensures maintaining high quality of the population[2, 3, 5, 12].

The mutation operator introduces random changes into chromosomes, thus expanding the area of the search space.

Repeated use of reduction, selection, crossover, and mutation operators contributes to improving the quality of each individual chromosome, and as a consequence, the population in general, reflecting the main goal of the genetic algorithm, i.e., improving the quality of the initial population. The main result of the genetic algorithm work is a chromosome of the final population on which the target function takes its extreme value.

Use of genetic algorithms adaptation in various subject areas

As mentioned above, the genetic algorithms are a strategic approach to the problem, which should be adapted to a certain subject area by setting parameters, and defining operators of the genetic algorithm. With that, the genetic algorithm becomes related to the subject area in question, and cannot be used for solving problems in another subject area[8-11].

Selection of parameters, operators and the type of chromosomes influences the search stability and speed, i.e., the main indicators of the genetic algorithm's efficiency. Speed is determined by the time required for achieving one of the stopping criteria stated above by the algorithm. Stability is the ability of a genetic algorithm to increase quality of the population, and to exceed local extrema.

For increasing speed, genetic algorithms may be subjected to multisequencing both at the level of organizing work of the algorithm, and at the level of its implementation in a computer.

At the level of work organization, multisequencing is performed due to structuring the population, which may be performed in two ways.

The first method is called the «islands concept», and consists in dividing the population into classes, the members of which cross only between themselves within the limits of the class, occasionally exchanging chromosomes on the basis of random sampling. The second method is called the "concept of crossover in the local area" and consists in setting a metric space on a population, the chromosomes of which are subjected to crossing only with their nearest neighbors[2,3,5].

Thus, given the possibility of parallel work of the genetic algorithm on a structured population, its presentation on structured Petri nets that also makes it possible to perform simulation modeling of the system's behavior becomes important[13-16].

In case of multisequencing at the level of implementation, both the stated above processes of crossing in pairs of parents, and the process of calculating the value of the efficiency function and using the mutation operator to chromosomes, may be implemented simultaneously on several subsystems within the integrated system, working in parallel. In describing the genetic algorithm based on the Petri nets theory, the task of their implementation on parallel Petri processors becomes important

and promising.

Stability of the search depends on parameters of the genetic algorithm operators.

For the crossover operator, such a parameter is the level of difference between children and parent chromosomes: the greater the difference is, the more stable the search is, but search speed is lower (the best result is achieved in longer time).

For the mutation operator, the parameter that influences stability of the search is the probability of its use: low probability ensures stable search and does not cause deterioration of chromosomes' quality.

The selection operator is related to search stability in the following way: constant selection of the strongest chromosomes usually causes convergence to the local extremum, while selection of weak chromosomes causes deterioration of the population's quality. The same is true for the reduction operator, too.

As to the influence of the population size on stability of the genetic algorithm, increased number of chromosomes in the population expands the search area; however one should reduce population to the initial size, otherwise speed of the genetic algorithm will decrease (similar algorithms are called generational)[8].

Development of generational algorithms led to emergence of adaptive genetic algorithms that changed their parameters during work. The nGA concept emerged, which represents multilevel genetic algorithms where the bottom level improves the population, and the top level optimizes parameters of the bottom one with regard to its speed and stability[11].

Using Petri nets for describing genetic algorithms

Thus, given the possibility of multilevel organization of a genetic algorithm, the task of its implementation in nested Petri nets becomes important.

Considering aforesaid, we can point out the following general properties of Petri nets[11, 13-16, 18-23] and of the genetic algorithm, justifying the advantage of using Petri nets for describing genetic algorithms.

1. Independence from the subject area;
2. Concurrency;
3. Probability and determinateness;
4. Structuredness;
5. Multilevel structure.

This paper will focus on solving the problem of structural synthesis of large discreet system with fixed structure (i.e., links between elements in large discreet systems with predetermined behavior cannot rebuild themselves in the process of processing the predefined reference vectors at the input and output of system) with the use of proposed methods.

RESULTS

Problem statement

Let us consider the following simplest class of tasks of structural synthesis of large discrete systems with predefined behavior.

Given:

$$S=<F,C> \quad ...(1)$$

where S is a large discrete system with predefined behavior, the synthesis of which should be performed, F = const is system structure, C = const is part of your system.

$$C= (C_1,...,C_R), \quad ...(2)$$

where Ci is the i-th component of the large discrete system, R is the number of components in the large discrete system.

$$C_i = \left\{ C_{ij} \right\}_{j=1}^{M_i}, \quad ...(3)$$

where C_{ij} is the j-th instance of the i-th component, Mi is the number of instances of the i-th component.

$$P = \left\{ P_k \right\}_{k=1}^{L} \quad ...(4)$$

where P is the number of properties that the synthesized large discrete system with predefined behavior can have, Pk is the k-th property of P_k, L is the number of properties of set P.

Required: for the given property , which the large discrete system with predefined behavior S should have, select one instance of each component Ci so that the large discrete system

$$S = \left\langle F, \left(C_{1j(k_0)}, \ldots, C_{Rj(k_0)} \right) \right\rangle \quad ...(5)$$

has property P_{k0}.

Problem solution

To solve this problem, let us use the genetic algorithm adapted to the solution with the help of nested Petri nets.

The genetic algorithm should be adapted to a certain subject area by setting parameters and defining operators of the genetic algorithm. In using genetic algorithms, alternative solutions are presented as a line of symbols with fixed links, and called a genotype. Therefore, special attention should be paid to formal description of the genotype.

In our case, the solution is a large discreet system S, definition (1)-(3) of which is presented in the problem statement. This definition shows that large discrete system S is described as a line of symbols (5). For obtaining description of large discreet discrete system S (1), its structure F and composition C in the language of Petri nets should be defined.

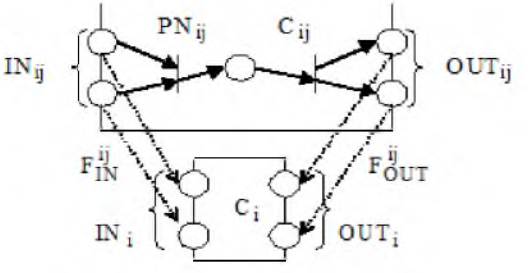

Fig. 1. Correspondence between component and instances

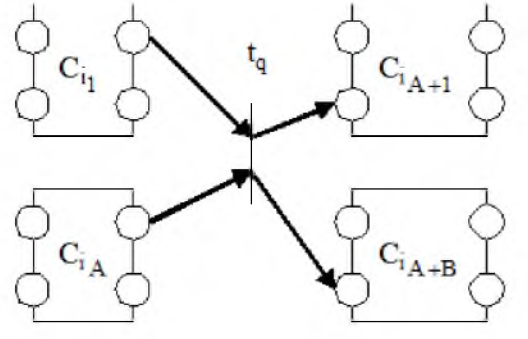

Fig. 2. Relationship between components

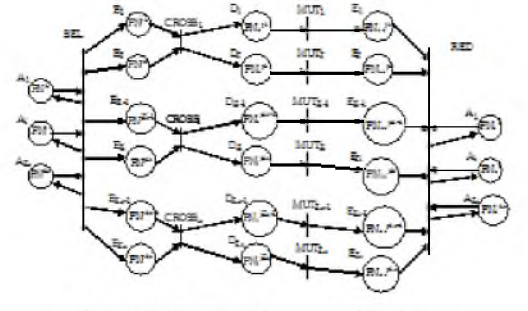

Fig. 3. Structure of a nested Petri net

From definitions (2) and (3) it follows that composition C is completely defined by the components, therefore each component Ci should be described as a Petri net, and this, in turn, means that each instance of components Ci should be defined in the language of Petri nets.

Let us consider an instance of Cij. Let us denote the Petri net modeling this instance as PNij. With regard to the fact that the instances of various components interact, each instance should be assigned inputs and outputs, which, naturally, would be modeled by positions of the PNij net. Let us denote the set of input and output positions as INij and OUTij, respectively.

Now let us consider component Ci. The structure of this component is fully determined by the instance of Cij, which represents it. The components interact via inputs and outputs that are simulated by the set of input and output positions, which would be denoted as INi and OUTi, respectively. Naturally, one-to-one correspondence $F^{ij}_{IN}: IN_{ij} \to IN_i$ and $F^{ij}_{OUT}: OUT_{ij} \to OUT_i$ should exist between elements of sets in IN_{ij} and IN_i, as well between OUTij and OUTi. All said above about components and instances may be illustrated in Figure 1.

Let us denote the Petri net that models component Ci as PNi. If component Ci is represented by instance then .

Finally, let us consider structure F of a large discreet system S. It is completely determined by interaction between the components, which is naturally simulated by transitions in Petri net. Let us denote the set of such transitions as T.

Every transition $t_q \in T$ connects output positions of some components C_{i1}, C_{i2}, …, C_{iA} with input positions of some other components C_{iA+1}, C_{iA+2}, … , C_{iA+B}, which is graphically illustrated in Figure 2.

correspondence $F: T \to \bigcup_{i=1}^{R} (IN_i \cup OUT_i)$ completely defines structure F of system S.

Thus, a Petri net modeling system S may be described as a line of code

$$PN = \langle PN_1, ..., PN_i, ..., PN_R, T, E \rangle,$$

where PN_i is model of component C_i, and set of transitions T and compliance E determine structure F of system S.

According to the problem statement, out of all possible models of system S, the one, which would have redefined behavior, should be selected. Reaction to input signals will be considered as system behavior. To do so, inputs and outputs should be selected, to be modeled by positions in the PN net. Let us define sets of input and output positions as IN and OUT, respectively.

Obviously, $IN \subset \bigcup_{i=1}^{R} IN_i$ and $OUT \subset \bigcup_{i=1}^{R} OUT_i$.

Behavior of system S will be a pair of non-negative integer vectors

$Z_{IN} = (z_1^{IN}, ..., z_{v_0}^{IN})$ and $Z_{OUT} = (z_1^{OUT}, ..., z_{W_0}^{OUT})$

where z_v^{IN} is the number of labels received at the V-th input position before starting the PN network , z_w^{OUT} is the number of labels received at the W-th output of position after stopping the PN network, and V_0 and W_0 are the numbers of elements of IN and OUT sets, respectively.

Consequently, set P of properties that the synthesized system may have is modeled by set $Z = \{Z_k\}_{k=1}^{L}$, where $Z_k = (Z_{IN}^k, Z_{OUT}^k)$ is the model of property P_k in form of a pair of non-negative integer vectors.

Thus, the task is reduced to the following. Out of all hypothetically possible models PN of system S, the one with Z_k property is to be found.

For checking whether the PN model has Z_k property, this model should be formed, its input should receive vector Z_{IN}^k, after which the PN net should be started; after the net is stopped, the number of labels at output OUT should be compared with vector Z_{OUT}^k.

The next step of the solution is defining the proximity measure for model PN to Z_k properties. To do so, first vector Z_{IN}^k should be supplied to input IN of the model, and vector Z_{OUT} should be received at the outlet. The proximity degree will be determined by the notion of metric space, and by treating the obtained vector Z_{OUT} and the reference vector Z_{OUT}^k as elements of Euclidean space $\Re^{W_0}$, i.e., a multitude of ordered sets of W_0 of real numbers $x = (x_1, ..., x_{W_0})$ with distance

..(6)

where $y = (y_1, ..., y_{W_0})$.

The lower is, the closer PN model to property Z_k is. Formula (6) may be extended for the case of a different value of coordinate-wise proximity of the obtained vector to the reference:

$$\rho_1(x, y) = \sum_{w=1}^{W_0} \alpha_w |x_w - y_w| \ , \ \alpha_1 + \alpha_2 + ... + \alpha_w = 1, \ \alpha_1, \ \alpha_2 , ..., \ \alpha_w \ e" 0$$

However, in this case additional problems occur, related to sensitivity of the distance to possible errors in defining weight coefficients $\alpha_1, \alpha_2, ..., \alpha_w$ (1). If $\rho_1(Z_{OUT}, Z_{OUT}^k) = 0$, then PN model has property Z_k. It is natural to consider distance ρ_1 as an efficiency function.

The operators of genetic algorithm manipulate the genotypes presented as a code line of symbols. In the theory of Petri nets, analog operators are transitions. It is therefore natural to simulate operators of a genetic algorithm by transitions in a Petri net. With regard to the fact that transitions manipulate Petri net labels, it is necessary to use such a Petri nets extension, where the labels can simulate a code line of symbols. Such an extension is represented by nested Petri nets (multilevel Petri nets or L-Petri nets). In our case, the code line is a Petri net. Therefore, we can use extension of nested Petri nets.

Thus, operators of a genetic algorithm will be modeled by transition of nested Petri net, and genotypes will be modeled by macro-labels.

Let us denote the initial population as

$G^0 = (PN^1, ...PN^{2n})$,

where PNi is the i-th model of the designed object in the form of a Petri net, 2n is fixed population

size. For the sake of simplicity of further description, it is assumed that the size of the population is an even number.

Let us place every PN^i model at a corresponding position Ai as macro-labels of a nested Petri net.

The selection operator is modeled by *SEL* transition, which will copy PN^i from position A^i and place these copies at positions B^1, B^2, ..., B^{2i-1}, B^{2i}, ..., B^{2n-1}, B^{2n} as a macro-labels. Let us denote the macro-labels at positions B^{2i-1} and B^{2i} as PN^{j2i-1} and PN^{j2i}. Macro labels are placed at positions B^1, ..., B^2n in accordance with the value of efficiency function (6) according to the rule: the less is the value of the function, the lower is the number of the position where the macro-label is placed. Thus, in position B^1 the best macro-label is placed, in position B^2 - a slightly worse macro-label, etc. up to position B^{2n}, where the worst macro-label will be placed.

The crossover operator is modeled by $CROSS^i$ transition, which will take macro-labels PN^{j2i-1} and PN^{j2i} from positions B^{2i-1} and B^{2i}, cross them over and place new macro-labels and into positions D^{2i-1} and D^{2i}, respectively. Macro-labels are crossed over according to the following rule:

1. From set {1,2,...,R}, a random number is selected, which we will denote as r.
2. Let us select the model of r-th component from genotypes PN^{j2i-1} and PN^{j2i}.

$$PN^{j2i-1} = \left\langle PN_1^{j2i-1}, ..., PN_r^{j2i-1}, ..., PN_R^{j2i-1}, T, F \right\rangle$$

$$PN^{j2i} = \left\langle PN_1^{j2i}, ..., PN_r^{j2i}, ..., PN_R^{j2i}, T, F \right\rangle$$

3. Swapping the selected models we obtain new macro-labels.

$$PN_*^{j2i-1} = \left\langle PN_1^{j2i-1}, ..., PN_r^{j2i}, ..., PN_R^{j2i-1}, T, F \right\rangle$$

$$PN_*^{j2i} = \left\langle PN_1^{j2i}, ..., PN_r^{j2i-1}, ..., PN_R^{j2i}, T, F \right\rangle$$

The mutation operator is modeled by a MUT_{2i} transition, which will take macro-label PN_*^{j2i} from position D_{2i}, perform its mutation and place the new macro-label PN_{**}^{j2i} into position E_{2i}. The macro-label undergoes mutation according to the following rule:

1. From set {1,2,...,R}, a random number is selected, which we will denote as r.
2. Let us select the model of r-th component from genotype.
3. From the set of models of instances of our r-th component, let us randomly select a new model, which we will denote as , and replace model with it, thus obtaining a new macro-label.

The operator of mutation for macro-label at position D2i-1 is modeled in a similar way.

The reduction operator is modeled by *RED* transition, which will take macro-labels from positions $E_1, ..., E_{2n}$ and from positions $A_1, ..., A_{2n}$, remove the worst macro-labels and place the remaining at positions $A_1, ..., A_{2n}$. Macro-labels are deleted according to the following rule:

1. Efficiency function (6) is calculated for each macro-label.
2. In position A1 the best macro-label is placed, in position A2 - a slightly worse macro-label, etc. up to position A2n, where the worst macro-label will be placed. Let us denote the macro-label at position A_t as PN_t*.
3. All other macro labels are deleted. Note that exactly 2_n macro labels will be deleted.

The structure of a nested Petri net with indication of all macro-labels is shown in Figure 3.

DISCUSSION

The work of the proposed model of a genetic algorithm adapted for solving the problem of structural synthesis with the use of nested Petri nets can be described in the following way:

1. For each component C_i, build model PN_{ij} for each of its instances C_{ij}.
2. Build model of structure F, defining the set of transitions T and correspondence E.
3. Define property P_k that a large discrete system with predefined behavior S should have, as a pair of vectors $Z_k = \left(Z_{IN}^k, Z_{OUT}^k\right)$.
4. Form initial population G^0.
5. Set the number of cycles for the work of a nested Petri net and/or conditions for its stop as the value of efficiency function ρ_1.
6. Place the initial population G^0 of the nested

proposed Petri net.

7. Start the nested Petri net.
8. After stopping the nested Petri net, position A will contain model PN of the large discrete system that would best satisfy the Z_k property.

The use of various parameters of the proposed procedure may lead to various solutions of the problem of synthesis, which may be united into a selection set, after which the final selection is performed by the decision-maker on the basis of his own (often non-formalized) preferences.

CONCLUSION

The main result of this work is formalization and algorithmization of the process of large discreet systems with predefined behavior by forming a solution with fixed structure out of predefined components.

Resolution of this problem is significantly complicated by the huge number of possible options of implementation, out of which it is necessary to choose the ones satisfying predefined behavior. Three theories were used for solving this problem: evolution methods, simulation modeling, and Petri nets. To overcome this problem, genetic algorithms adapted for resolving this problem by using 2-level nested Petri nets were used.

The proposed approach will further be used for structural synthesis of large discrete systems with predefined behavior and dynamic inter-component relations. This suggests working of the genetic algorithm not only with components of the synthesized system, but with relations between elements, too. This complicates solving of the problem with the use of the existing methods, since the search area significantly increases. For solving more complex problem, next works will use an individual layer of inter-element relations - "inter-element bus". It will make it possible to optimize the process of system synthesis with the use of genetic algorithm on the base of nested Petri nets.

In theory, it seems promising to extend the proposed approach to the tasks of structural and parametric synthesis of large discrete systems with changing structure and composition of the used elements.

REFERENCES

1. Lomazov, V.A., V.I. Lomazova and D.A. Petrosov, Evolutionary procedure of decision-making support in modeling of inter-related processes. Problems of Modern Science and Practice (University n.a. V. I. Vernadsky), 2014; **2**(51): 82-89.http://elibrary.ru/item.asp?id=21623772 http://elibrary.ru/contents.asp?issueid=1271899 http://elibrary.ru/contents.asp? issueid =1271899 &selid=21623772
2. Semenkin, Å.S., Evolutionary strategies algorithm based approaches for the linear dynamic system identification. In Adaptive and Natural Computing Algorithms. Lecture Notes in Computer Science, Volume 7824. – Springer-Verlag, Berlin, Heidelberg, 2013; 477-484.
3. Emelyanov, V.V., V.V. Kureichik and V.M. Kureichik, Theory and practice of evolutionary modeling. Moscow: Fizmatlit, 2003; 432.
4. Sibani, P. and H.J. Jensen, Stochastic Dynamics of Complex Systems. World Scientific and Imperial College Press, 2013; 300. ISBN 978-1-84816-993-7https://en.wikipedia.org/w/index.php?title=Paolo_Sibani&action=edit&redlink=1http://www.worldscientific.com/worldscibooks/10.1142/p877https://en.wikipedia.org/wiki/Special:BookSources/9781848169937https://en.wikipedia.org/wiki/World_Scientifichttps://en.wikipedia.org/wiki/Imperial_College_Press
5. Kureichik, V.M., B.K. Lebedev and O.K. Lebedev, Search adaptation: theory and practice. Moscow: Fizmatlit, 2006; 272.
6. Gladkov, L.A., V.V. Kureichikand and V.M. Kureichik, Genetic algorithms: Tutorial, 2nd ed. Moscow: Fizmatlit, 2006; 320.
7. Eiben, A.E. et al, "Genetic algorithms with multi-parent recombination". In the Proceedings of the International Conference on Evolutionary Computation: PPSN III. The Third Conference on Parallel Problem Solving from Nature, 1994; 78–87.
8. Hingston, Ph., L. Barone and Z. Michalewicz, Design by Evolution: Advances in Evolutionary Design. Springer, 2008. ISBN 978-3540741091. https://en.wikipedia.org/wiki/International_Standard_Book_Numberhttps://en.wikipedia.org/wiki/Special:BookSources/978-3540741091
9. Schmitt, L.M., Theory of Genetic Algorithms. *Theoretical Computer Science,* 2001; **259**: 1–61.
10. Schmitt, L.M., Theory of Genetic Algorithms II: models for genetic operators over the string-tensor representation of populations and convergence

to global optima for arbitrary fitness function under scaling. *Theoretical Computer Science*, 2004; **310**: 181–231.

11. Akbari, Z., "A multilevel evolutionary algorithm for optimizing numerical functions". IJIEC, 2010; **2**(2011): 419–430.
12. Poli, R., W.B. Langdon and N.F. McPhee, 2008. A Field Guide to Genetic Programming, pp: 250. Date views: 12.02.15, Lulu.com. ISBN 978-1-4092-0073-4https://ru.wikipedia. org/wiki/ %D0%A1%D0%BB%D1 %83%D0 %B6% D0%B5%D0%B 1%D0%BD%D0% B0% D1%8F:%D0%98% D1%81% D1%82%D0% BE %D1%87%D0%BD%D 0%B8%D0 %BA%D0%B8_% D0 %BA%D0%BD%D0 %B8%D0%B3/9781409200734
13. Peterson, J., The theory of Petri nets and systems modeling. Moscow: Mir, 1984; 264.
14. Kotov, V.E., Petri Nets. Moscow: Nauka, 1984; 160.
15. Dawis, E.P., J.F. Dawis, and Wei-Pin Koo, 2001. Architecture of Computer-based Systems using Dualistic Petri Nets. In the Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics,3: 1554–1558.
16. Magergut, V.Z. and A.A. Klimov, Adaptive approach to developing algorithms of logical devices work marked with Petri nets. In the Proceedings of the XXIII Int. Scientific Conf. "Mathematical Methods in Technics and Technologies" (MMTT-23, Saratov: SSTU), 2010; **10**: 17-20.
17. Lomazov, V.A., V.L. Mikhailova, D.A. Petrosov and D.B. Elchaninov, Evolutionary procedure of structural and parametric synthesis of imitation models for document flow systems. Scientific Bulletin of the Belgorod state University. Series: History. Political science. Economics. Informatics, 2013; **28(1)**: 204.http://elibrary.ru/contents.asp ?titleid=28240
18. Marakhovsky, V.B., L.Ya. Rosenblum and A.V. Yakovlev, Modeling of concurrent processes. Petri nets. A course for system architects, programmers, system analysts, and designers of complex management systems. St. Petersburg: Professional literature, IT training, 2014; 400. http://profliteratura.ru/index.php?option =com_ content&view=article&id=77:modelirovanie-parallelnykh-protsessov&catid=84&Itemid=645.
19. Ignatenko, V.A. and V.Z. Magergut, 2010.An information Petri net as a basis for fuzzy objects control / V. A. Ignatenko. In the Proceedings of the XXIII Int. Scientific Conf. "Mathematical Methods in Technics and Technologies" (MMTT-23, Saratov: SSTU), 10: 13-17.
20. Ignatenko, V.A. and V.Z. Magergut, Parallel processing of management algorithms using information Petri nets. In the Proceedings of the XXIV International Scientific Conference: Mathematical Methods in Technics and Technologies (in 10 volumes Section 6,7.Eds., Balakirev, V.S., Kyiv: National. Tech. University of Ukraine "KPI"), 2011; **6**: 73-75.
22. Ignatenko, V.A. and V.Z. Magergut, Describing dynamic processes with the use of an information Petri net. Scientific Bulletin of BSU. History, *Political Science, Economics, Computer Science,* 2011; **13**: 161-179.
23. Ignatenko, V.A. and V.Z. Magergut, An information Petri net as a tool for parallel processing of control algorithms. Scientific Bulletin of BSU. History, Political Science, *Economics, Computer Science*, 2011; **19**: 119-126.