DOI: 10.15587/978-617-8360-13-9.CH2

Oleksandr Gaman, Svitlana Kashkevich,
Oleh Shknai, Andrii Radchenko, Serhii Neronov, Zaur Musaiev

CHAPTER 2

METHODS OF PROCESSING VARIOUS DATA IN INTELLIGENT SYSTEMS FOR MANAGEMENT OF THE NETWORK AND SERVER ARCHITECTURE OF THE INTERNET OF COMBAT THINGS

ABSTRACT

In this chapter of the research, a method of processing various types of data in intelligent management systems of the network and server architecture of the internet of military equipment is proposed.

The basis of this research is the theory of artificial intelligence, namely evolving artificial neural networks, basic genetic algorithm procedures, neuro-fuzzy expert systems, as well as bio-inspired algorithms.

In the course of the research, the authors proposed:

– a complex model of processing various types of data in intelligent decision-making support systems;

– a method of processing various types of data in intelligent management systems of network and server architecture;

– a method of increasing the efficiency of processing various types of data in intelligent management systems of network and server architecture.

The use of methods of processing various types of data in intelligent management systems of the network and server architecture of the internet of military equipment:

– to reduce the probability of premature convergence of the metaheuristic algorithm while processing various types of data in intelligent management systems of the network and server architecture of the internet of military equipment;

– to maintain a balance between the speed of convergence of the metaheuristic algorithm and diversification while processing various types of data in the intelligent control systems of the network and server architecture of the internet of military equipment;

– to take into account the type of uncertainty and noise of the data of the metaheuristic algorithm while processing various types of data in the intelligent control systems of the network and server architecture of the internet of military equipment;

– to take into account the available computing resources of the system while processing various types of data in the intelligent management systems of the network and server architecture of the internet of military equipment;

– to take into account the priority of search by swarm agents of the meta-heuristic algorithm while processing various types of data in intelligent management systems of the network and server architecture of the internet of military equipment;

– to conduct an initial display of flock individuals taking into account the type of uncertainty;

– to conduct accurate training of metaheuristic algorithms;

– to conduct a local and global search taking into account the degree of data noise while processing various types of data in intelligent management systems of the network and server architecture of the internet of military equipment.

KEYWORDS

Internet of military equipment, processing of various types of data, combined systems, reliability and efficiency.

2.1 THE DEVELOPMENT OF A COMPLEX MODEL FOR PROCESSING VARIOUS DATA

The development of IoT and its adaptation on a large scale will require both new theoretical research and significant funding. The distribution of network resources of various technical means will require new approaches and intelligent automation. An important issue will be the perception of the amount of information that IoT will provide.

The most promising option for increasing the efficiency (and, as a consequence, the efficiency) of information processing is the use of approaches based on artificial intelligence. One such toolkit of artificial intelligence is the expert system.

Currently, expert systems have become the main tool used to solve various types of tasks (interpretation, forecast, diagnosis, planning, design, control, debugging, instruction and management) in a wide variety of problem areas [1–9]. The functioning of expert systems is based on the knowledge model [10–18]. It contains a set of principles that describe the state and behavior of the research object. The most widely used knowledge model for expert systems is the production model due to its simplicity, ease of processing and comprehensibility of the end user [19–32].

However, recently, fuzzy expert systems have become widespread. This type of expert systems is based on a set of rules that use linguistic variables and fuzzy relations to describe the state and behavior of the object under investigation [33–54].

The rules presented in this form are the closest to natural language, so there is no need to use a separate expert knowledge engineer to create and edit the rules. In most cases, they can be edited by the expert itself practically without special training [55–67].

One of the most important problems inherent in knowledge-based systems is the problem of knowledge representation [65–71]. This is explained by the fact that the form of knowledge representation significantly affects the characteristics and properties of the system. In order to operate all kinds of knowledge from the real world with the help of a personal computer, it is necessary to carry out their simulation.

In such cases, it is necessary to distinguish knowledge intended for processing by computing devices from knowledge used by humans. In addition, with a large amount of knowledge, it is desirable to simplify the sequential management of individual elements of knowledge.

Taking into account the above, the aim of the research is to develop a complex model of processing various types of data in intelligent decision-making support systems. The object of research is the intelligent decision-making support system. The subject of research is the knowledge representation model in intelligent decision-making support systems.

On the basis of the decision-making support system model [6], as well as the model of a complex task, which is the processing of various types of data in intelligent server and network architecture management systems, let's build a model of intelligent DMSS:

$$DMSS = < PRT, prt^{dm}, R^{dm}, M^{dm} >, \tag{2.1}$$

where $PRT = \{prt_q \mid q = 1,\dots,N_{prt}\}$ is a set of expert models; prt^{dm} is a model of a decision maker; $R^{dm} = \{r^{dmq} \mid q = 1,\dots,N_{prt}\}$ is the relationship between the decision maker and experts, for example, the relationship of information exchange; M^{dm} are the methods of processing information received from experts.

Each expert works strictly in its field of expertise $S_{prtq} \in S$, where S is the set of all fields of knowledge necessary to solve the task of processing various types of information and does not deal with any subtasks other than its own, $S_q \cap S_w = \varnothing$, with $q, w = 1,\dots,N_{prt}$; $q \neq w$.

Based on considerations from research [5] and taking into account that in real tasks subtasks are solved by experts step by step, the expert model can be presented:

$$prt_q = < B_{prof}, B_{theor}, B_{prec}, B_{facts}, MET_{prtq}, S_{prtq}, In_{prtq}, \Delta t >, \tag{2.2}$$

where B_{prof} is the production base of professional knowledge; B_{theor} is the production base of theoretical knowledge; B_{prec} is the basis of precedents (experience); B_{facts} is the basis of facts; MET_{prtq} is a set of reasoning methods; S_{prtq} is the description of the expert's field of knowledge; In_{prtq} is an interpreter that ensures the execution of a sequence of rules for solving a task based on facts and rules stored in databases and knowledge; Δt is the period of issuing intermediate decisions by experts.

The model of the decision maker can be built by analogy with (2.2):

$$prt^{dm} = < B_{prof}, B_{theor}, B_{prec}, B_{facts}, B_{ext}, MET_{prldm}, S_{prldm}, In_{prldm}, E, T >, \tag{2.3}$$

where B_{ext} is the production knowledge base on how to perform reduction, aggregation, comparison and coordination; E is a set of coordinating processes; T is the processing time of various types of data.

The expression (2.3) in comparison with expression (2.2) has significant differences. Production knowledge base B_{ext} about how the decision maker manages the processing of various types of data. This knowledge is obtained from other experts.

The set E describes how the decision maker can coordinate the work of experts, B_{prof} is base of professional knowledge, B_{theor} is the basis of theoretical knowledge, B_{prec} is the basis of precedents (experience).

Let's consider how the intelligent DMSS functions according to the work (2.1). Let the decision maker be given a task prb^u, which is reduced to subtasks $prb_1^h,...,prb_{N_h}^h$. Analyzing the expression (2.1) and (2.3), as well as relying on the practice of processing various types of data, it is possible to draw the following conclusions: GL^h contained in B_{pred} and B_{facts} is the experience combined with facts allows the expert to determine what result should be obtained; MET^h contained in $B_{prof}, B_{theor}, B_{pred}, MET_{prti}, S_{prti}, In_{prti}$; DAT^h contained in B_{facts}.

In traditional DMSS, described, for example, in work [5], each expert prt_q, $q = 1,...,N_{prt}$, having received its subtask of processing various types of data prb_j^h, $j = 1,...,N_h$, finds its solution using its professional skills B_{prof} and theoretical knowledge B_{theor}.

After finishing the process of processing various types of data, it issues a result $sol^{hj} \in SOL_j^h$, where SOL_j^h is a set of results of solving the task prb_j^h, which can be written as a correspondence ψ_4:

$$\psi_4: DAT^h \otimes B^U \to SOL^h,\ B^U = B_{prof} \cup B_{theor}. \tag{2.4}$$

Compliance elements ψ_4 are the tuples $\left(\left(\left\{dat_\sigma^h\right\},\{b_\beta^u\}\right),sol_\gamma^h\right)$, with $\sigma = 1,...,N_{dath}$; $\beta = 1,...,N_b$; $\gamma = 1,...,N_{sh}$, where the first component is a two-component vector consisting of a list of initial data $\left\{dat_\sigma^h\right\}$, $dat_\sigma^h \in DAT^h$ and list of knowledge $\{b_\beta^u\}$, $b_\beta^u \in B^U$ (professional knowledge are production rules; theoretical knowledge are analytical dependencies) and the second is the result $sol_\gamma^h \in SOL^h$ processing of various types of data prb^h.

Conformity ψ_4 is not a function (cannot be written analytically or calculated by numerical methods) because the knowledge of an expert and the results of processing an element of heterogeneous data can be represented in natural language. It is ambiguous, because with an incomplete set of initial data (a priori uncertainty), the expert can offer several options for results, subjectively, because each decision on the processing of different types of data prb^h corresponds to at least one element of $DAT^h \otimes B^U$ and not injunctive, because not every element of $DAT^h \otimes B^U$ corresponds to the solution of the task prb^h.

Let's indicate the number of stages into which experts divide the process of solving partial tasks N_{sol} and sol_l^h is the result of solving a partial task at the l-th stage, $l = 1,...,N_{sol}$. A time interval is allocated to the stage of processing various types of data Δt. Because in practical tasks, the total time T for solving the task of processing various types of data prb^u, strictly limited and time Δt between refinements of the task of processing various types of data is constant, the number of stages is determined by the formula:

$$N_{sol} = T/\Delta t. \tag{2.5}$$

It should be noted that in the process of solving a partial task of processing various types of data prb^h, through the coordinating influences of the decision maker, the original data DAT^h in expression (2.4) can be modified – additional information is entered or outdated information is replaced with new information.

Let DAT_l^h output data for the l-th stage, $l=1,\dots,N_{sol}$. Then DAT_1^h is the initial data received from the decision maker, moreover $DAT_1^h = DAT^h$ and DAT_l^h, $l=2,\dots,N_{sol}$ is the output data of the following stages.

The index l means the number of the stage in which the raw data is used. Let's define DAT_{l+1}^h as initial data of the $l+1$-th stage, obtained after the coordinating influences of the decision maker regarding the change of the data of the l-th stage.

The scheme of the sequence of stages of the expert's work in finding a solution to a partial task π^h can be expressed as follows:

$$DAT_l^h \otimes B^U \otimes \left\{sol_1^h\right\} \otimes,\dots,\otimes \left\{sol_{l-1}^h\right\} \Rightarrow \left\{sol_1^h\right\},\ l=1,\dots,N_{sol}. \tag{2.6}$$

Output data DAT_l^h, $l=1,\dots,N_{sol}$ at each stage are supplemented by coordinating influences $e^\alpha \in E$, issued by the decision maker to the expert, which are determined on the basis of the integrated result of solving the task of processing various types of data prb^u on $l-1$-th stage. Let's assume that the expert is given one coordinating influence of one type. Let's determine the correspondence ψ_5:

$$\psi_5 : \left\{sol_l^u\right\} \otimes B_{ext} \to E,\ l=1,\dots,N_{sol}-1. \tag{2.7}$$

The maximum value of l is equal to $N_{sol}-1$ because after N_{sol} stage, it is no longer possible to use coordination, since the final result is obtained.

sol_l^u is the integrated result of solving the task of processing various types of data prb^u at the l-th stage; $E=\{\varepsilon_1,\dots,\varepsilon_{N_\varepsilon}\}$ is a set of type vectors $\left(e_1^1,\dots,e_{N_{prt}}^6\right)$, each component of which is a coordinating action for the expert, $e_q^\alpha \in E, q=1,\dots,N_{prt}$.

Since the knowledge of integration is included B_{ext} the decision maker (4) is the integrated result sol_l^u solving the complex task of processing various types of data prb^u can be written like this:

$$\left\{sol_l^{h1}\right\} \otimes \dots \otimes \left\{sol_l^{hN_h}\right\} \otimes B_{ext} \to \left\{sol_l^u\right\}, \tag{2.8}$$

where $sol_l^{h1},\dots,sol_l^{hN_h}$ are the solving partial tasks for processing various types of data $prb_1^h,\dots,prb_{N_h}^h$ in accordance.

Compliance elements ψ_5 are the tuples $\left(\left(sol_l^u,\{b_{ext}^\mu\}\right),\varepsilon_p\right)$, with $l=1,\dots,N_{sol}$, $\mu=1,\dots,\mu_{N_\mu}$, $p=1,\dots,N_\varepsilon$, where the first component is a two-component vector consisting of the integrated result sol_l^u solving the task of processing various types of data prb^u at the l-th stage and the list of knowledge used by the decision maker on how to perform the comparison

$\left\{b^{\mu}_{ext}\right\}$, $b^{\mu}_{ext} \in B_{ext}$ and the second component is a vector whose elements are coordinating actions $e \in E$ for an expert.

On N_{sol}-th stage ($l = N_{sol}$) vector of coordinating influences $\varepsilon = \left(e_1^{\alpha},\ldots,e_{N_{prt}}^{\alpha}\right)$, $\alpha = 6$, thus, the decision maker does not issue coordinating influences to experts, but only aggregates (integrates solutions to partial tasks prt^h into a single, integrated solution sol_l^u of a difficult task prb^u) the results of their work.

If the integrated result is obtained sol_l^u and does not suit the decision maker, it must revise the original tasks prb^u, so change DAT_l^h for prb^h or change the list of your knowledge B_{ext} and expert knowledge B_{prof} and after that, initiate the re-operation of the intelligent DMSS.

In accordance, ψ_5 is not a function (cannot be written analytically and calculated), since the knowledge of the decision maker and the integrated result of solving the task prb^u can be represented in natural language. It is unambiguous, as each expert is assigned a specific coordinating action e_q^{α} and therefore compliance ψ_5 uniquely defines only one vector $\varepsilon \in E$. It is subjective, because for each vector $\varepsilon \in E$ at least one element matches $\left\{sol_l^u\right\} \otimes B_{ext}$ and not inductively, because not to every element $\left\{sol_l^u\right\} \otimes B_{ext}$ corresponds to a vector $\varepsilon \in E$.

The choice of whether the intermediate result of the solution of a partial task of processing various types of data prb_w^h, $w = 1,\ldots,N_h$ is ready and expert $prt_q \in PRT$, $q = 1,\ldots,N_{prt}$ accepts based on experience B_{prec} (2.3).

The functional multi-agent system model presented in work [4] is adopted as the basic model for processing homogeneous data in intelligent DMSS. The choice of this model is explained by the fact that the complexity of the homogeneous tasks to be solved is not great – there are relatively few input and output parameters and expert rules, so there is no need to investigate the micro-level of subtasks.

It can also be noted that the decomposition of the task of processing heterogeneous data into a set of tasks of processing homogeneous data is transparent, it is not difficult to establish a connection between subtasks and elements of the intelligent DMSS.

The conceptual model of an element of a multi-agent intelligent DMSS for processing homogeneous data can be presented:

$$res^e = R_1^{resmet}\left(res^e, met^e\right) \circ R_1^{respr}\left(res^e, pr^{ei}\right) \circ R_1^{respr}\left(res^e, pr^{eo}\right) \circ R_1^{resst}\left(res^e, st^e\right) \circ$$
$$\circ R_1^{stst}\left(st^e(t), st^e(t+1)\right) \circ R_1^{prst}\left(pr^{ei}(t), st^e(t+1)\right) \circ R_1^{stpr}\left(st^e(t), pr^{eo}(t)\right), \qquad (2.9)$$

where R_1^{stst}, R_1^{prst}, R_1^{stpr} is the relationship "state – state", "input – state", "state – output", respectively.

Among the crowd $MET^e = \{met_y^e \mid y = 1,\ldots,N_{met}\}$ autonomous methods will be highlighted met_1^e are analytical calculations, met_2^e are neurocomputing, met_3^e are fuzzy calculations, met_4^e is reasoning based on experience; met_5^e are evolutionary calculations, met_6^e are statistical calculations, met_7^e is logical reasoning. If between element res^e and autonomous method met_y^e relationship is established $R_1^{resmet}\left(res^e, met_y^e\right)$, let's denote the element res^{ey}. The appropriate calculation method is selected according to expression (2.1).

Relations R_1^{prpr}, R_2^{prpr} (2.9) are given on sets of variables DAT^u, GL^u and sets of variables DAT^h, GL^h from the processing of homogeneous data included in the task of processing heterogeneous data.

The following cases are possible:

1) a set of variables for prb^u coincides with the set of variables for prb^h, so $DAT^u = DAT^h$, $GL^u = GL^h$;

2) a set of variables for prb^h is a subset of the corresponding set u_{prb}, so $DAT^h \subset DAT^u$, $GL^h \subset GL^u$;

3) a set of variables of a subset of the corresponding set prb^h, so $DAT^u \subset DAT^h$, $GL^u \subset GL^h$. The second variant is typical for the analyzed task of processing heterogeneous data, because during the solution of the task of processing heterogeneous data, the initial data is divided between different partial tasks of processing homogeneous data.

The extension of model (2.9) is performed based on the following considerations. In the process of coordination, the intermediate states of the solution of partial tasks for the processing of homogeneous data are controlled.

In the accepted notation (2.9), these states mean the states (decision results) of functional elements res^e, simulating the solutions of partial tasks for processing homogeneous data prb^h. From the analysis of these states, the "input" properties change during coordination pr^{ei} of one or more elements res^e.

To take this fact into account, let's introduce a triple into the conceptual model (2.9). $R_1^{stpr}\left(st^u(t), pr^{ui}(t+1)\right)$. In other words, based on the state of intellectual DMSS $st^u(t)$ at time t, the output data changes $pr^{ui}(t+1)$ for intelligent DMSS, but already at the moment of time $t+1$, so for the next iteration.

Plural R_1^{stpr} establishes relationships between state $st^u(t)$ of hybrid res_A^u (2.9) at the moment of the model time t and the state of the inputs of one or more elements res^e in the next step. To make the necessary change of inputs pr^{ei} of one or more functional elements res^e (2.9), let's enter a triple $R_1^{stact}\left(st^u, act^{ek}\right)$, where $ACT^{ek} = \left\{act_1^{ek\alpha}, \ldots, act_{N_{prt}}^{ek\alpha}\right\}$ is the set of concepts denoting coordinating actions, which is identical to the set of coordinating actions E, where α is the type of coordinating action, $\alpha = 1, \ldots, 6$. In the coordination algorithm, the coordinating actions are described by the knowledge base B_{ext}. Plural R_1^{stact} is a set of relations between states st^u of intellectual support system res_A^u at the moment of the model time t and the necessary coordinating actions ACT^{ek}.

The modified conceptual model of the intelligent DMSS for processing homogeneous data with coordination has the following form:

$$res_A^u = res_A^u \circ R_1^{stpr}\left(st^u(t), pr^{ui}(t+1)\right), \qquad (2.10)$$

and a modified model of an element of an intelligent DMSS for processing homogeneous data:

$$res^e = res^e \circ R_1^{stact}\left(st^u, act^{ek}\right). \qquad (2.11)$$

Relations R_1^{stpr} and R_1^{stact} are not set in advance, like R_1^{stst}, R_1^{prst}, R_1^{stpr} are fixed during the operation of the intelligent DMSS for processing homogeneous data and are the result of the solution of the k-task prb^k.

The following is a conceptual model of the operation of an intelligent DMSS for processing homogeneous data, built according to the expressions (2.10), (2.11):

$$st^{ek}(t_0') \Rightarrow \begin{Bmatrix} st^{e1}(t_0) \\ st^{e2}(t_0) \end{Bmatrix} \to \begin{Bmatrix} st^{e1}(t_1) \\ st^{e2}(t_1) \end{Bmatrix} \Rightarrow st^{ek}(t_0') \to st^{ek}(t_1') \Rightarrow$$
$$\Rightarrow \begin{Bmatrix} st^{e1}(t_1) \\ st^{e2}(t_1) \end{Bmatrix} \to \begin{Bmatrix} st^{e1}(t_2) \\ st^{e2}(t_2) \end{Bmatrix} \Rightarrow st^{ek}(t_1') \to st^{ek}(t_2') \Rightarrow \ldots \Rightarrow$$
$$\Rightarrow \begin{Bmatrix} st^{e1}(t_{p-1}) \\ st^{e2}(t_{p-1}) \end{Bmatrix} \to \begin{Bmatrix} st^{e1}(t_p) \\ st^{e2}(t_p) \end{Bmatrix} \Rightarrow st^{ek}(t_{p-1}') \to st^{ek}(t_p'), \tag{2.12}$$

where "$\Rightarrow$" is the relationship R^{stst}, which connect the states from different subspaces and specify the transition from one homogeneous space to others during the functioning of the intelligent DMSS for processing homogeneous data; "$\to$" is the transition between states within the corresponding subspace.

Transitions "$\Rightarrow$" from the element subspace res_k^{e7} model the issue of coordinating influences from the decision maker to experts while processing homogeneous data. And the set of transitions "$\Rightarrow$" and "$\to$" allows to simulate and trace the process of self-organization in the process of the operation of an intelligent DMSS for the processing of homogeneous data.

In the expression (2.12), curly brackets indicate the beginning and end of the parallel work of the functional elements of the intelligent DMSS for processing homogeneous data. It can be seen from the model that after each fixation " $\}\Rightarrow$" of states, functional elements res_q^{ey} control of the element res_k^{e7} is transferred and after it changes its state, control is transferred to a group of functional elements.

This model is related to the conceptual model:

$$\begin{Bmatrix} st^{e1}(t) \to st^{e1}(t+1) \to \ldots \to st^{e1}(t+n) \\ st^{e2}(t) \to st^{e2}(t+1) \to \ldots \to st^{e2}(t+n) \end{Bmatrix}. \tag{2.13}$$

The simulation of the proposed model was carried out in the MathCad 2014 software environment for evaluating the radioelectronic environment.

The initial setting of the membership functions of the set of terms of the neuron fuzzy expert system has been performed, since all sources of radio radiation have different characteristics. The experts indicated which values of primary and calculated parameters are considered high for radio emission devices, which are average and which are low. The membership functions for the analysis of the radio-electronic situation are presented in the specified form according to the formula:

1) (PJ="H") and (KOV="H") and (UN="H") and (PW="L") $\rightarrow$ *(BER="H"),*

...

81) (PJ="L") and (KOV="L") and (UN="L") and (PW="H") $\rightarrow$ *(BER="L"),*

82) (FJ="L") and (FW="L") and (UN="H") and (PW="H") $\rightarrow$ *(BER="N"),*

...

108) (SC="L") and (KOV="L") and (UN="H") and (PJ="L") $\rightarrow$ *(BER="N"),*

In this example, part of the rule base of the neuro-fuzzy expert system is given. In the main base of rules there are rules not only with connections of conditions with the help of *T*-norms, but also with the help of *T*-conorms and with negations of conditions.

In the worst case, to find a solution, the system should check all the rules contained in the rule base. Thus, it is necessary to check 405 conditions and calculate 297 *T*-norm operations. This is an unacceptably long process, given the limitations of the hardware.

The input data for the neuro-fuzzy expert system are indicators of the power of transmitters of radio-emitting devices, the type of signal-code structures, the uncertainty of the radio-electronic situation, the frequency of radiation of radio-emitting devices. After passing through the phase of fuzzification, the system received fuzzy estimates for each monitored parameter.

The results of the assessment of the state of the radio-electronic situation for various presentation models are shown in **Fig. 2.1**.

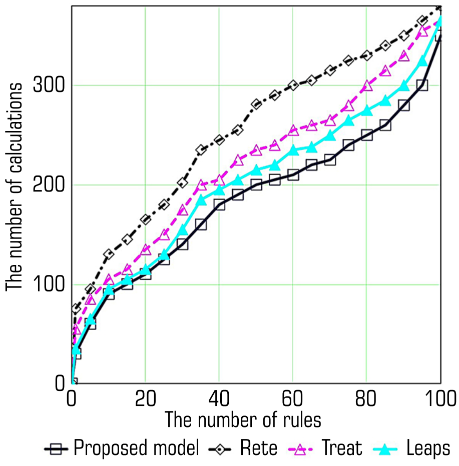

Fig. 2.1 Comparison of the efficiency of the obtained estimate for different models

Features of radio-electronic situation analysis systems are: a large number of analyzed parameters; dynamic change of the electronic environment; functioning in conditions of uncertainty about the state of the radio-electronic situation; constant updating of the signal database; functioning under the influence of natural and intentional disturbances.

As a result of the conducted research, the following was established:

– a complex task of processing heterogeneous data under certain restrictions and assumptions can be divided (performed decomposition) into a number of separate partial tasks of processing homogeneous data;

– taking into account the specifics of the data that must be processed in intelligent DMSS, the availability of computing power, it is necessary to carefully approach the use of mathematical apparatus for modeling and processing of data circulating in the specified systems;

– with homogeneous data, it is advisable to use the model described by expressions (9)–(12) and after adding new types of data, it is advisable to increase the models of the lower level.

The direction of further research should be the improvement of information processing methods in intelligent decision-making support systems.

2.2 THE METHOD OF PROCESSING VARIOUS TYPES OF DATA IN INTELLIGENT MANAGEMENT SYSTEMS OF NETWORK AND SERVER ARCHITECTURE

This method is based on the duck flocking algorithm (DFA). There are two rules that must be followed in the foraging process of duck agents (DA). Rule One: While searching for food, DA with strong search ability are positioned closer to the center of the food source, which attracts other DA to move closer to them. The updated location is also affected by nearby DA. Rule two: while looking for food, all DA approach food. The next position is affected by the adjacent DA and the food position or the DA leader. The method of processing various types of data in intelligent management systems of network and server architecture consists of the following sequence of actions:

Step 1. Input of initial data. At this stage, the raw data available on the intelligent network and server architecture management system are entered.

The main parameters of DFA are the maximum number of iterations $Iter_{max}$, the population size NP, the number of decision variables n, the probability of the presence of a predator Pdp, the scaling factor sf, the migration constant G_c, as well as the upper and lower bounds for the decision variable FS_U and FS_L:

$$FS_{i,j} = \left(FS_L + \left(rand(\ )\right) * \iota\right) * \left(FS_U - FS_L\right), \tag{2.14}$$

$$i = 1, 2, \ldots, NP,\ j = 1, 2, \ldots, n,$$

where $rand()$ is a uniformly distributed random number in the range [0, 1]; ι is a correction coefficient that determines the degree of a priori information about the state intelligent network and server architecture management system.

Step 2. Numbering of DA in the flock, i, $i \in [0, S]$. At this stage, each DA is assigned a serial number.

Let's add an identifier for each duck in the Duck structure so that each agent has its own unique number.

Step 3. Determination of the initial speed of DA. The initial speed v_0 of each DA is determined by the following expression:

$$v_i = (v_1, v_2, \ldots, v_S),\ v_i = v_0. \tag{2.15}$$

Step 4. Exhibition of DA on the search plane.

At this stage, the DA is issued taking into account the type of uncertainty about the state intelligent network and server architecture management system, and the basic model of its state is initialized [2].

Let's suppose that the expression for a randomly generated initial position in a *D*-dimensional search space is:

$$X_i = L_b + (U_b - L_b) \cdot o, \tag{2.16}$$

where X_i is the spatial position of the *i*-th DA ($i=1, 2, 3, \ldots, N$) in the DA group; N is the population size; L_b and U_b represent the upper and lower bounds of the search space, respectively; o is a matrix of random numbers between (0, 1).

Step 5. Search for food sources by individual DA individuals.

After taking turns, the DA flock arrived at a location with more food. Each DA gradually disperses and begins to search for food, this process is defined as follows:

$$X_i^{t+1} = \begin{cases} X_i^t + \mu \cdot X_i^t \cdot sign(r - 0.5),\ P > rand; \\ X_i^t + CF_1 \cdot (X_{leader}^t - X_i^t) + CF_2 \cdot (X_j^t - X_i^t),\ P \le rand, \end{cases} \tag{2.17}$$

where $sign(r-0.5)$ affects the process of finding food and it can be set to -1 or 1; μ is the global search control parameter; P is the probability of conversion of the search of the reconnaissance phase; CF_1 and CF_2 are the coefficients of cooperation and competition between DA at the search stage, respectively; X_{leader}^t is the best position of the DA of the current historical value in the *t*-th iteration; X_j^t is the DA around X_i^t in the search for feeding the DA group in the *t*-th iteration.

Parameter μ can be calculated as follows:

$$\mu = K \cdot (1 - t/t_{\max}), \tag{2.18}$$

where K is calculated by the formula:

$$K = \sin(2 \cdot rand) + 1. \tag{2.19}$$

The search range of a flock of ducks is wider when $\mu > 1$. This non-linear strategy is used to improve the global search capability of the proposed gas station. In addition, this phase can also integrate two parameters C_1 and C_2 will be factored into the update formula used to balance DA positions in the research phase.

The update formula is described as follows:

$$X_i^{t+1} = X_i^t + C_1 \cdot \mu \cdot \left(X_i^t - C_2 \cdot X_{leader}^t\right) \cdot sign\left(r - 0.5\right) + \\ + CF_1 \cdot \left(X_{leader}^t - X_i^t\right) + CF_2 \cdot \left(X_j^t - X_i^t\right), \tag{2.20}$$

where μ is the control parameter of the global DA search; C_1 and C_2 are the parameters for adjusting the balance of the position; C_1 is a random number in (0, 1). C_2 can be set not only as random, but also as a constant, for example 0, 1 or others; CF_1 and CF_2 are the coefficients of cooperation and competition between DA at the search stage; X_{leader}^t is the best position of the DA of the current historical value in the t-th iteration; X_j^t is the number of individuals around X_i^t in the search for food by the DA group in the t-th iteration.

Step 6. Classification of food sources for DA.

Locations of the best DA food source (minimum fitness) are considered watercress (FS_{ht}), locations from the following three food sources have vegetables (FS_{at}) and the rest are considered common plants (FS_{nt}):

$$FS_{ht}=FS(\text{sorte\_index}(1)), \tag{2.21}$$

$$FS_{at}(1{:}3)=FS(\text{sorte\_index}(2{:}4)), \tag{2.22}$$

$$FS_{nt}(1{:}NP\text{-}4)=FS(\text{sorte\_index}(5{:}NP)). \tag{2.23}$$

Step 7. Sorting of the best DA individuals. The selection of the best individuals is carried out using the improved genetic algorithm proposed in work [13]. While searching for food, the strongest DA, which is the largest in the flock, directs the other DA in the group to look for food. The weight in DA is similar to the quality of the objective function values of the candidate solutions. Therefore, the best solution variant with the best value for the objective function is considered to be the largest DA in the group. This search behavior of DA leads to different scanning areas of the search space, which improves the exploration ability of DA in global search.

Step 8. The search for food sources by the DA flock.

After that, a flock of ducks collects enough food to meet its own nutritional needs. This process is closely related to the suitability of the position of each DA and is defined as follows:

$$X_i^{t+1} = \begin{cases} X_i^t + \mu \cdot \left(X_{leader}^t - X_i^t\right), f\left(X_i^t\right) > f\left(X_i^{t+1}\right); \\ X_i^t + KF_1 \cdot \left(X_{leader}^t - X_i^t\right) + KF_2 \cdot \left(X_k^t - X_j^t\right), \text{ otherwise,} \end{cases} \tag{2.24}$$

where μ is the controlling parameter of the global search at the stage of the group food search of DA; KF_1 and KF_2 are the coefficients of cooperation and competition between DA at the stage of group foraging of DA; X^t_{leader} is the best position of the DA of the current historical value in the t-th iteration; X^t_k and X^t_j are the numbers of DA around X^t_i in the search for food by the DA group in the t-th iteration, where $k \neq j$.

All parameter values CF_1, CF_2, KF_1 and KF_2 are in the range of values (0, 2), and the calculation formula can be summarized as follows:

$$CF_i \text{ or } KF_i \leftarrow \frac{1}{FP} \cdot rand(0,1) \; (i = 1, 2), \tag{2.25}$$

where FP is a constant, it is set to 0.618; $rand$ is a random number in works (0, 1).

Step 9. Checking the presence of a predator. At this stage, AM is checked for the presence of predators. If there are predators, go to Step 8. If there are no predators, the end of calculations.

End of the algorithm.

The software implementation of the method of processing various types of data in intelligent management systems of network and server architecture is given below:

```
// Duck Swarm Algorithm.cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <limits>
#include <cstdlib>
#include <ctime>
using namespace std;
const int POPULATION_SIZE = 50;
const int DIMENSIONS = 2;
const int MAX_ITERATIONS = 1000;
const double INERTIA = 0.5;
const double LEARNING_RATE = 2.0;
struct Duck {
   vector<double> position;
   vector<double> velocity;
   double fitness;
   Duck(int dimensions) {
      position.resize(dimensions);
      velocity.resize(dimensions);
      fitness = numeric_limits<double>::infinity();}};
double fitnessFunction(const vector<double>& position) {
    double sum = 0.0;
    for (double x : position) {
        sum += x * x;}
    return sum;}
void updatePosition(Duck& duck, const vector<double>& bestPosition) {
    for (int i = 0; i < DIMENSIONS; ++i) {
        duck.velocity[i] = INERTIA * duck.velocity[i] +
            LEARNING_RATE * ((double)rand() / RAND_MAX) * (bestPosition[i] – duck.position[i]);
        duck.position[i] += duck.velocity[i]; }}
void duckSwarmOptimization() {
    vector<Duck> population(POPULATION_SIZE, Duck(DIMENSIONS));
    vector<double> globalBestPosition(DIMENSIONS);
    double globalBestFitness = numeric_limits<double>::infinity();
    for (Duck& duck : population) {
        for (int i = 0; i < DIMENSIONS; ++i) {
            duck.position[i] = ((double)rand() / RAND_MAX) * 10 – 5;
            duck.velocity[i] = ((double)rand() / RAND_MAX) * 2 – 1;  }
        duck.fitness = fitnessFunction(duck.position);
        if (duck.fitness < globalBestFitness) {
            globalBestFitness = duck.fitness;
            globalBestPosition = duck.position;
        } }
    for (int iter = 0; iter < MAX_ITERATIONS; ++iter) {
        for (Duck& duck : population) {
            updatePosition(duck, globalBestPosition);
            duck.fitness = fitnessFunction(duck.position);
            if (duck.fitness < globalBestFitness) {
                globalBestFitness = duck.fitness;
                globalBestPosition = duck.position; }}
      cout << "Iteration " << iter + 1 << " Best decision: " << globalBestFitness << endl;}
    cout << " Optimized solution: ";
    for (double x : globalBestPosition) {
        cout << x << " "; }
    cout << endl;}
int main() {
    srand(time(0));
    duckSwarmOptimization();
    return 0;
}
```

2.3 THE METHOD OF INCREASING THE EFFICIENCY OF PROCESSING VARIOUS TYPES OF DATA IN INTELLIGENT MANAGEMENT SYSTEMS OF NETWORK AND SERVER ARCHITECTURE

In this chapter of the dissertation research, an optimizer based on the simulation of reptile behavior (the case of crocodiles and alligators) is proposed – a population-based stochastic algorithm that uses reptilian agents (RA) as search agents.

The method of increasing the efficiency of processing various types of data in intelligent management systems of network and server architecture consists of the following sequence of actions:

Step 1. Input of initial data. At this stage, the main parameters of the algorithm are determined, such as:

– the type of task being solved;

– the number of agents in the population;

– the type of data (structured, unstructured), archived, real-time data;

– the number of variables characterizing the task being solved;

– available computing resources of the system;

– the type of uncertainty about the hierarchical system (complete uncertainty, partial uncertainty, complete awareness);

– the volume and type of the research sample;

– the volume and type of test sample;

– artificial neural network architecture, etc.

Step 2. Creation of a RA flock. Initialization of the RA population X_i $(i=1, 2, \ldots, n)$ takes place. RA set form a population described by the matrix X. The initial population of RA in this set of algorithms is generated taking into account the uncertainty about the state of the intelligent management system of the network and server architecture based on the constraints of the problem under consideration. Members of the RA population are search agents in the solution space, providing candidate values for the problem variables based on their positions in the search space. Mathematically, each member of the general population is a vector, the number of elements of which is equal to the number of task variables.

The issuance of RA is carried out taking into account uncertainty about the data circulating in the system based on the basic system model and circulating data models [2, 19, 21]:

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1}\times\iota_{1,1} & \cdots & x_{1,d}\times\iota_{1,d} & \cdots & x_{1,m}\times\iota_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1}\times\iota_{i,1} & \cdots & x_{i,d}\times\iota_{i,d} & \cdots & x_{i,m}\times\iota_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N,1}\times\iota_{N,1} & \cdots & x_{N,d}\times\iota_{N,d} & \cdots & x_{N,m}\times\iota_{N,m} \end{bmatrix}_{N\times m}, \qquad (2.26)$$

where X is the population matrix of RA; X_i is the i-th member of the RA flock (solution candidate); $x_{i,d}$ is the d-th dimension in the search space (decision variable); N is the number of RA; m is the number of decision variables.

Step 3. Numbering of RA in the flock, i, $i \in [0, S]$. At this stage, each RA is assigned a serial number. This makes it possible to determine the parameters of finding a solution for each individual in the flock.

Step 4. Determination of the initial speed of the RA.

Initial speed v_0 of each RA is determined by the following expression:

$$v_i = \left(v_1, v_2, \ldots, v_S\right), v_i = v_0. \tag{2.27}$$

The process of updating the RA population is based on the simulation of two strategies: the exploration phase and the exploitation phase.

Step 5. Preliminary assessment of the RA search area. In this procedure, the natural language search section is determined precisely by the halo of RA existence. Given that food sources for RA are food of animal origin, it is advisable to sort the suitability of food sources (Step 6).

Step 6. Classification of food sources for RA.

The location of the best food source (minimum fitness) is considered to be (FS_{ht}) the animal food (carrion) that is nearby and requires the least energy expenditure to find and obtain it. Delicacy food of animal origin will be denoted as FS_{at}.

Other non-priority food sources (food that is necessary for the survival of individuals) will be designated as FS_{nt}:

$$FS_{ht} = FS(\text{sorte\_index}(1)), \tag{2.28}$$

$$FS_{at}(1:3) = FS(\text{sorte\_index}(2:4)), \tag{2.29}$$

$$FS_{nt}(1:NP-4) = FS(\text{sorte\_index}(5:NP)). \tag{2.30}$$

Step 7. Determination of the number of available computing resources of the system.

At this stage, the amount of computing resources available for calculations is determined. In accordance with the provisions outlined in Step 4, the concept of updating the provisions of the RA is chosen.

Step 8. Intelligence (surrounding the prey).

The encirclement (reconnaissance) phase is the phase of global exploration of RA space. The RA research strategy is related to the current number of iterations. If $t \leq 0.25T$, the RA will enter the strategy of a high move. The reconnaissance phase is described by the following mathematical expression:

$$Xnew_i^j = \begin{cases} XG^j \times -\eta_i^j \times \beta - R_i^j \times (\varsigma \cdot rand), \; t \leq \dfrac{T}{4}; \\ XG^j \times X_{r1}^j \times (\varsigma \cdot rand) \times ES, \; t > \dfrac{T}{4} \text{ and } t \leq \dfrac{T}{2}, \end{cases} \tag{2.31}$$

where $Xnew_i^j$ is the j-th dimension of the i-th new RA solution; XG^j is the j-th dimension of the optimal solution obtained at the moment; t is the current iteration number; T is the maximum number of iterations; η_i^j is the j-th hunting operator of the i-th RA, which is calculated using the expression (2.32); β is a constant and is equal to 0.1; R_i^j is the value of the i-th option of the decision, used to reduce the search area by the j-th size, which is calculated using equation (2.33); r_1 randomly takes values from 1 to n; ς is the degree of noise of the data circulating in the system; ES is a random number in the range from –2 to 2, when the number of iterations decreases and equation (2.34) will be used to calculate:

$$\eta_i^j = XG^j \times P_i^j, \tag{2.32}$$

$$R_i^j = \frac{XG^j - X_{r2}^j}{XG^j - \varepsilon}, \tag{2.33}$$

$$ES = 2 \times r_3 \times \left(1 - \frac{1}{T}\right), \tag{2.34}$$

$$P_i^j = \alpha + \frac{X_i^j - Md_i}{XG^j \times \left(UB^j - LB^j\right) + \varepsilon}, \tag{2.35}$$

In equation (2.32), ε is the minimum; r_2 is a number chosen randomly from the range 1–n. In equation (2.33), r_3 is a random integer from –1 to 1. In equation (2.34), $\alpha=0.1$; Md_i is the average position of the i-th candidate for the solution, which is calculated using equation (2.35):

$$Md_i = \frac{1}{d}\sum_{j=1}^{d} X_i^j, \tag{2.36}$$

where d is the dimensionality of the problem to be solved.

Step 9. Verification of hitting the global optimum. At this stage, the condition for the algorithm to reach the global optimum is checked according to the specified optimization criterion.

Step 10. Global restart procedure.

The restart procedure can effectively improve the ability of the algorithm to go beyond the current optimum and improve the exploratory ability of the algorithm. If the optimal population of the algorithm remains unchanged after *ke* iterations, the population will most likely collapse into a local optimum. Thus, the candidate solution will be initialized randomly to speed up the departure from the global optimum:

$$Xnew_i^j = rand \times \left(UB - LB\right) + LB,\ nt > ke. \tag{2.37}$$

Step 11. Hunting phase (exploitation).

Under the condition $t>0.5T$, the hunting phase begins, and when $t>3/4T$ and $t\leq T$ are the hunting cooperation strategies of the RA. Its equation for updating position is as follows:

$$Xnew_i^j = \begin{cases} XG^j \times P_i^j \times (\varsigma \cdot rand), \ t > \frac{T}{2} \text{ and } t \le 3\frac{T}{4}; \\ XG^j \times \eta_i^j \times \varepsilon - R_{r1}^j \times (\varsigma \cdot rand), \ t > 3\frac{T}{4} \text{ and } t \le T. \end{cases} \quad (2.38)$$

Finally, if the position of the new candidate RA is closer to the food than the current one, the RA will move to the new candidate position and proceed to the next iteration:

$$X_i^j(t+1) = Xnew_i^j(t), \text{ if } F(X_i(t)) > F(Xnew_i(t)), \quad (2.39)$$

where *F*() is a function to calculate the matching value; X_i is the location of the *i*-th candidate solution; $Xnew_i$ is the location of the *i*-th new candidate solution.

Step 12. Combining individual optimization algorithms into a mixed one.

To combine different types of natural optimization algorithms, an ensemble mutation strategy is used, which can generate various individuals to improve the global search capabilities of the hybrid algorithm, which is written as follows:

$$V_{i1} = \begin{cases} X_{R1} + F_1 \times (X_{R2} - X_{R3}), \ r_{10} < C_1; \\ X_i, \ r_{10} \ge C_1, \end{cases} \quad (2.40)$$

$$V_{i2} = \begin{cases} X_{R4} + F_2 \times (X_{R5} - X_{R6}) + F_2 \times (X_{R7} - X_{R8}), \ r_{11} < C_2; \\ X_i, \ r_{11} \ge C_2, \end{cases} \quad (2.41)$$

$$V_{i3} = \begin{cases} X_i + F_3 \times (X_{R9} - X_i) + F_3 \times (X_{R10} - X_{R11}), \ r_{12} < C_3; \\ X_i, \ r_{12} \ge C_3, \end{cases} \quad (2.42)$$

where V_{i1}, V_{i2} and V_{i3} are newly generated mutant positions of the *i*-th search agent; $R_1 \sim R_{11}$ are the different integer indicators in the range [1, *N*]; F_1, F_2 and F_3 are scale factors with values of 1.0, 0.8, and 1.0, respectively; $r_{10} \sim r_{12}$ are the random numbers in the range [0, 1]. In addition, the C_1, C_2 and C_3 parameters are 0.1, 0.2, and 0.9, indicating the crossover rate.

After generating candidate mutant positions V_{i1}, V_{i2} and V_{i3}, the best position V_i with the lowest fitness value will be selected to compare with the fitness of the original position X_i, and then the best position will be saved as a new X_i to participate in the next iteration calculation. These processes can be described using equation (2.43):

$$X_i = \begin{cases} V_i, & \text{if } F(V_i) < F(X_i); \\ X_i, & \text{otherwise}, \end{cases} \quad (2.43)$$

where *F*(·) is the cost function.

Step 13. Checking the stop criterion. The algorithm terminates if the maximum number of iterations is completed. Otherwise, the behavior of generating new places and checking conditions is repeated.

Step 14. Learning RA knowledge bases.

In this research, the learning method based on evolving artificial neural networks developed in the research [2] is used to learn the knowledge bases of each RA.

The method is used to change the nature of movement of each RA, for more accurate analysis results in the future.

Learning the RA knowledge base is an important step that allows agents to accumulate experience and improve their performance in the process of performing tasks. This learning may include storing information about discovered resources, objective functions, optimal routes, the behavior of other agents, and mechanisms that help agents make decisions based on the received data.

The end of the algorithm.

The software implementation of the method of increasing the efficiency of processing various types of data in intelligent management systems of network and server architecture is given below:

```
// reptilian agents.cpp
#include <iostream>
#include <vector>
#include <unordered_map>
#include <thread>
#include <future>
#include <mutex>
#include <cstdlib>
#include <ctime>
class Agent {
public:
   int id;
   double speed;
   double position;
   double bestPosition;
   double bestValue;
   Agent(int id, double speed)
      : id(id), speed(speed), position(0.0), bestPosition(0.0), bestValue(-1.0) {}
   void move() {
      position += speed; }
   void updateBest(double value) {
      if (value > bestValue) {
         bestValue = value;
         bestPosition = position; } }};
class AgentSwarm {private:
```

```
std::vector<Agent> agents;
std::mutex mtx;
double globalBestValue;
double globalBestPosition; public:
AgentSwarm(int numAgents) : globalBestValue(-1.0), globalBestPosition(0.0) {
   std::srand(static_cast<unsigned int>(std::time(nullptr)));
   for (int i = 0; i < numAgents; ++i) {
      double speed = (std::rand() % 10) + 1;
      agents.emplace_back(i, speed); }}
void moveAgents() {
   std::vector<std::future<void>> futures;
   for (auto& agent : agents) {
      futures.emplace_back(std::async(std::launch::async, [&]() {
         std::lock_guard<std::mutex> lock(mtx);
         agent.move();
         double value = evaluate(agent.position);
         agent.updateBest(value);
         updateGlobalBest(agent);
         std::cout << "Agent " << agent.id << " moved to position: " << agent.position
            << ", Value: " << value << std::endl;}));}
   for (auto& future : futures) {
      future.get(); }}
double evaluate(double position) {
   return -1 * (position – 5) * (position – 5) + 10; }
void updateGlobalBest(Agent& agent) {
   if (agent.bestValue > globalBestValue) {
      globalBestValue = agent.bestValue;
      globalBestPosition = agent.bestPosition; }}
void checkGlobalOptimum() {
   std::cout << "Global Best Position: " << globalBestPosition
      << ", Global Best Value: " << globalBestValue << std::endl; }
void globalRestart() {
   std::cout << "Restarting swarm..." << std::endl;
   for (auto& agent : agents) {
      agent.position = 0.0;
      agent.bestPosition = 0.0;
      agent.bestValue = -1.0; }
   globalBestValue = -1.0;
   globalBestPosition = 0.0;}
```

```
        void huntPhase() {
            std::cout << "Hunting Phase Started..." << std::endl;
            moveAgents();  }
        void learnKnowledgeBase() {
            std::cout << "Learning Knowledge Base..." << std::endl; }};
    class FoodSource {
    public:
        std::string type;
        double value;
        FoodSource(std::string type, double value) : type(type), value(value) {}};
    void classifyFoodSources() {
        std::vector<FoodSource> foodSources = {
            FoodSource("Insect", 10.0),
            FoodSource("Plant", 5.0),
            FoodSource("Meat", 20.0)};
        std::cout << "Food sources classified:\n";
        for (const auto& food : foodSources) {
            std::cout << "Type: " << food.type << ", Value: " << food.value << std::endl; }}
    int main() {
        int numAgents = 5;
        std::cout << "Number of agents: " << numAgents << std::endl;
        AgentSwarm swarm(numAgents);
        std::cout << "Available computational resources: " << std::thread::hardware_concurrency()
    << " threads" << std::endl;
        classifyFoodSources();
        for (int i = 0; i < 3; ++i) {
            std::cout << "\nIteration " << i + 1 << std::endl;
            swarm.moveAgents();
            swarm.checkGlobalOptimum();
            swarm.huntPhase();
            swarm.learnKnowledgeBase();
            swarm.globalRestart(); }
        return 0;}
```

CONCLUSIONS

1. The research proposed a complex model of processing various types of data in intelligent decision-making support systems.

The essence of the complex approach in modeling is that two partial models are proposed: a model for processing heterogeneous data in intelligent decision-making support systems and a model for processing homogeneous data in intelligent decision-making support systems.

The analysis of the model of the intelligent DMSS for the processing of various types of data allows to draw the following conclusion. While solving the task of processing various types of data, a model of intelligent DMSS is proposed, in contrast to the traditional ones, even in the process of solving partial tasks incorrectly started by experts prb^h with the help of self-organization, expressed in the coordination of partial tasks, the decision maker strives for an ideal solution to the task of processing various types of data. This increases the efficiency of finding an acceptable result from the processing of various types of data prb^u. At the same time, errors in the processing of various types of data will be detected and corrected before the result is obtained prb^u. At the same time, in classic DMSS, a repeated solution is required to detect an error in the processing of various types of data.

The model of homogeneous data processing is based on the idea that the same processing of homogeneous data in intelligent DMSS can be solved in parallel by different functional elements. The relations of integration of elements arise as internal non-verbal images in the memory of the user, who can compare the dynamics of the simulation of the task of processing various types of data in intelligent DMSS from different points of view, which allows to see what modeling using a single model does not provide. Another assumption is developed in the model: the inclusion of a model of a person making a decision in the model of the intelligent DMSS leads to the emergence of the effect of self-organization. This makes it possible to interrelate the results of the work of individual functional elements of the intelligent DMSS in the process of synthesizing the solution to the task of processing various types of data, and not after, as in known models, which ensures the relevance of the intelligent DMSS to the real process of collective problem solving.

2. In this research, a method of processing various types of data in intelligent management systems of network and server architecture is proposed. The essence of the method consists in the processing of various types of data, which are different in nature, with units of measurement due to the use of the duck flock algorithm.

The novelties of the proposed method are:

– the type of uncertainty of the initial data on the state of the intelligent management system of the network and server architecture is taken into account, thereby reducing the time for making the first decision;

– the presence of a procedure for the implementation of an adaptive strategy for the search for food sources of the DA, which allows determining the priority of information processing;

– taking into account the presence of a predator while choosing food sources, which increases the level of information security during its processing;

– taking into account the available computing resources of the intelligent network and server architecture management system, thereby preventing overloading of the intelligent network and server architecture management system.

3. Also, in this research, a method of increasing the efficiency of processing various types of data in intelligent management systems of network and server architecture is proposed. The essence of the method of increasing the efficiency of processing various types of data is to reduce the time of processing various types of data due to the use of an improved algorithm of the herd of reptiles.

The novelties of the proposed method are:

– taking into account the type of uncertainty and noisy data circulating in intelligent network and server architecture management systems. This is achieved through the use of appropriate correction coefficients;

– using the procedure of ensemble mutation, which allows to increase the efficiency of processing various types of data due to the use of the selection of appropriate metaheuristic algorithms and their joint data use;

– changing the search area by individual agents, which is achieved by adjusting the arrival of various types of data;

– changing the speed of movement of agents, which achieves the priority of processing various types of data;

– conducting training of knowledge bases, which is carried out by training the synaptic weights of the artificial neural network, the type and parameters of the membership function, as well as the architecture of individual elements and the architecture of the artificial neural network as a whole;

– avoiding the problem of local extremum.

REFERENCES

1. Dudnyk, V., Sinenko, Y., Matsyk, M., Demchenko, Y., Zhyvotovskyi, R., Repilo, I. et al. (2020). Development of a method for training artificial neural networks for intelligent decision support systems. Eastern-European Journal of Enterprise Technologies, 3 (2 (105)), 37–47. https://doi.org/10.15587/1729-4061.2020.203301
2. Sova, O., Shyshatskyi, A., Salnikova, O., Zhuk, O., Trotsko, O., Hrokholskyi, Y. (2021). Development of a method for assessment and forecasting of the radio electronic environment. EUREKA: Physics and Engineering, 4, 30–40. https://doi.org/10.21303/2461-4262.2021.001940
3. Pievtsov, H., Turinskyi, O., Zhyvotovskyi, R., Sova, O., Zvieriev, O., Lanetskii, B., Shyshatskyi, A. (2020). Development of an advanced method of finding solutions for neuro-fuzzy expert systems of analysis of the radioelectronic situation. EUREKA: Physics and Engineering, 4, 78–89. https://doi.org/10.21303/2461-4262.2020.001353
4. Zuiev, P., Zhyvotovskyi, R., Zvieriev, O., Hatsenko, S., Kuprii, V., Nakonechnyi, O. et al. (2020). Development of complex methodology of processing heterogeneous data in intelligent decision support systems. Eastern-European Journal of Enterprise Technologies, 4 (9 (106)), 14–23. https://doi.org/10.15587/1729-4061.2020.208554

5. Kuchuk, N., Mohammed, A. S., Shyshatskyi, A., Nalapko, O. (2019). The Method of Improving the Efficiency of Routes Selection in Networks of Connection with the Possibility of Self-Organization. International Journal of Advanced Trends in Computer Science and Engineering, 8 (1.2), 1–6.
6. Shyshatskyi, A., Zvieriev, O., Salnikova, O., Demchenko, Ye., Trotsko, O., Neroznak, Ye. (2020). Complex Methods of Processing Different Data in Intellectual Systems for Decision Support System. International Journal of Advanced Trends in Computer Science and Engineering, 9 (4), 5583–5590. https://doi.org/10.30534/ijatcse/2020/206942020
7. Pozna, C., Precup, R.-E., Horvath, E., Petriu, E. M. (2022). Hybrid Particle Filter–Particle Swarm Optimization Algorithm and Application to Fuzzy Controlled Servo Systems. IEEE Transactions on Fuzzy Systems, 30 (10), 4286–4297. https://doi.org/10.1109/tfuzz.2022.3146986
8. Yang, X.-S., Deb, S. (2013). Cuckoo search: recent advances and applications. Neural Computing and Applications, 24 (1), 169–174. https://doi.org/10.1007/s00521-013-1367-1
9. Mirjalili, S. (2015). The Ant Lion Optimizer. Advances in Engineering Software, 83, 80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010
10. Yu, J. J. Q., Li, V. O. K. (2015). A social spider algorithm for global optimization. Applied Soft Computing, 30, 614–627. https://doi.org/10.1016/j.asoc.2015.02.014
11. Mirjalili, S., Mirjalili, S. M., Lewis, A. (2014). Grey Wolf Optimizer. Advances in Engineering Software, 69, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
12. Koval, V., Nechyporuk, O., Shyshatskyi, A., Nalapko, O., Shknai, O., Zhyvylo, Y. et al. (2023). Improvement of the optimization method based on the cat pack algorithm. Eastern-European Journal of Enterprise Technologies, 1 (9 (121)), 41–48. https://doi.org/10.15587/1729-4061.2023.273786
13. Gupta, E., Saxena, A. (2015). Robust generation control strategy based on Grey Wolf Optimizer. Journal of Electrical Systems, 11, 174–188.
14. Chaman-Motlagh, A. (2015). Superdefect Photonic Crystal Filter Optimization Using Grey Wolf Optimizer. IEEE Photonics Technology Letters, 27 (22), 2355–2358. https://doi.org/10.1109/lpt.2015.2464332
15. Nuaekaew, K., Artrit, P., Pholdee, N., Bureerat, S. (2017). Optimal reactive power dispatch problem using a two-archive multi-objective grey wolf optimizer. Expert Systems with Applications, 87, 79–89. https://doi.org/10.1016/j.eswa.2017.06.009
16. Koval, M., Sova, O., Orlov, O., Shyshatskyi, A., Artabaiev, Y., Shknai, O. et al. (2022). Improvement of complex resource management of special-purpose communication systems. Eastern-European Journal of Enterprise Technologies, 5 (9 (119)), 34–44. https://doi.org/10.15587/1729-4061.2022.266009
17. Ali, M., El-Hameed, M. A., Farahat, M. A. (2017). Effective parameters' identification for polymer electrolyte membrane fuel cell models using grey wolf optimizer. Renewable Energy, 111, 455–462. https://doi.org/10.1016/j.renene.2017.04.036
18. Zhang, S., Zhou, Y. (2017). Template matching using grey wolf optimizer with lateral inhibition. Optik, 130, 1229–1243. https://doi.org/10.1016/j.ijleo.2016.11.173

19. Khouni, S. E., Menacer, T. (2023). Nizar optimization algorithm: a novel metaheuristic algorithm for global optimization and engineering applications. The Journal of Supercomputing, 80 (3), 3229–3281. https://doi.org/10.1007/s11227-023-05579-4
20. Saremi, S., Mirjalili, S., Lewis, A. (2017). Grasshopper Optimisation Algorithm: Theory and application. Advances in Engineering Software, 105, 30–47. https://doi.org/10.1016/j.advengsoft.2017.01.004
21. Braik, M. S. (2021). Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems. Expert Systems with Applications, 174, 114685. https://doi.org/10.1016/j.eswa.2021.114685
22. Thamer, K. A., Sova, O., Shaposhnikova, O., Yashchenok, V., Stanovska, I., Shostak, S. et al. (2024). Development of a solution search method using a combined bio-inspired algorithm. Eastern-European Journal of Enterprise Technologies, 1 (4 (127)), 6–13. https://doi.org/10.15587/1729-4061.2024.298205
23. Yapici, H., Cetinkaya, N. (2019). A new meta-heuristic optimizer: Pathfinder algorithm. Applied Soft Computing, 78, 545–568. https://doi.org/10.1016/j.asoc.2019.03.012
24. Duan, H., Qiao, P. (2014). Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning. International Journal of Intelligent Computing and Cybernetics, 7 (1), 24–37. https://doi.org/10.1108/ijicc-02-2014-0005
25. Shyshatskyi, A., Romanov, O., Shknai, O., Babenko, V., Koshlan, O., Pluhina, T. et al. (2023). Development of a solution search method using the improved emperor penguin algorithm. Eastern-European Journal of Enterprise Technologies, 6 (4 (126)), 6–13. https://doi.org/10.15587/1729-4061.2023.291008
26. Yang, X.-S. (2012). Flower Pollination Algorithm for Global Optimization. Unconventional Computation and Natural Computation, 240–249. https://doi.org/10.1007/978-3-642-32894-7_27
27. Gomes, G. F., da Cunha, S. S., Ancelotti, A. C. (2018). A sunflower optimization (SFO) algorithm applied to damage identification on laminated composite plates. Engineering with Computers, 35 (2), 619–626. https://doi.org/10.1007/s00366-018-0620-8
28. Mehrabian, A. R., Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization. Ecological Informatics, 1 (4), 355–366. https://doi.org/10.1016/j.ecoinf.2006.07.003
29. Qi, X., Zhu, Y., Chen, H., Zhang, D., Niu, B. (2013). An Idea Based on Plant Root Growth for Numerical Optimization. Intelligent Computing Theories and Technology. Berlin: Heidelberg, 571–578. https://doi.org/10.1007/978-3-642-39482-9_66
30. Bezuhlyi, V., Oliynyk, V., Romanenko, I., Zhuk, O., Kuzavkov, V., Borysov, O. et al. (2021). Development of object state estimation method in intelligent decision support systems. Eastern-European Journal of Enterprise Technologies, 5 (3 (113)), 54–64. https://doi.org/10.15587/1729-4061.2021.239854
31. Mahdi, Q. A., Shyshatskyi, A., Prokopenko, Y., Ivakhnenko, T., Kupriyenko, D., Golian, V. et al. (2021). Development of estimation and forecasting method in intelligent decision support

systems. Eastern-European Journal of Enterprise Technologies, 3 (9 (111)), 51–62. https://doi.org/10.15587/1729-4061.2021.232718

32. Sova, O., Radzivilov, H., Shyshatskyi, A., Shevchenko, D., Molodetskyi, B., Stryhun, V. et al. (2022). Development of the method of increasing the efficiency of information transfer in the special purpose networks. Eastern-European Journal of Enterprise Technologies, 3 (4 (117)), 6–14. https://doi.org/10.15587/1729-4061.2022.259727
33. Zhang, H., Zhu, Y., Chen, H. (2013). Root growth model: a novel approach to numerical function optimization and simulation of plant root system. Soft Computing, 18 (3), 521–537. https://doi.org/10.1007/s00500-013-1073-z
34. Labbi, Y., Attous, D. B., Gabbar, H. A., Mahdad, B., Zidan, A. (2016). A new rooted tree optimization algorithm for economic dispatch with valve-point effect. International Journal of Electrical Power & Energy Systems, 79, 298–311. https://doi.org/10.1016/j.ijepes.2016.01.028
35. Murase, H. (2000). Finite element inverse analysis using a photosynthetic algorithm. Computers and Electronics in Agriculture, 29 (1-2), 115–123. https://doi.org/10.1016/s0168-1699(00)00139-3
36. Zhao, S., Zhang, T., Ma, S., Chen, M. (2022). Dandelion Optimizer: A nature-inspired metaheuristic algorithm for engineering applications. Engineering Applications of Artificial Intelligence, 114, 105075. https://doi.org/10.1016/j.engappai.2022.105075
37. Paliwal, N., Srivastava, L., Pandit, M. (2020). Application of grey wolf optimization algorithm for load frequency control in multi-source single area power system. Evolutionary Intelligence, 15 (1), 563–584. https://doi.org/10.1007/s12065-020-00530-5
38. Dorigo, M., Blum, C. (2005). Ant colony optimization theory: A survey. Theoretical Computer Science, 344 (2-3), 243–278. https://doi.org/10.1016/j.tcs.2005.05.020
39. Poli, R., Kennedy, J., Blackwell, T. (2007). Particle swarm optimization. Swarm Intelligence, 1 (1), 33–57. https://doi.org/10.1007/s11721-007-0002-0
40. Bansal, J. C., Sharma, H., Jadon, S. S., Clerc, M. (2014). Spider Monkey Optimization algorithm for numerical optimization. Memetic Computing, 6 (1), 31–47. https://doi.org/10.1007/s12293-013-0128-0
41. Yeromina, N., Kurban, V., Mykus, S., Peredrii, O., Voloshchenko, O., Kosenko, V. et al. (2021). The Creation of the Database for Mobile Robots Navigation under the Conditions of Flexible Change of Flight Assignment. International Journal of Emerging Technology and Advanced Engineering, 11 (5), 37–44. https://doi.org/10.46338/ijetae0521_05
42. Maccarone, A. D., Brzorad, J. N., Stone, H. M. (2008). Characteristics And Energetics of Great Egret and Snowy Egret Foraging Flights. Waterbird, 31 (4), 541–549. https://doi.org/10.1675/1524-4695-31.4.541
43. Ramaji, I. J., Memari, A. M. (2018). Interpretation of structural analytical models from the coordination view in building information models. Automation in Construction, 90, 117–133. https://doi.org/10.1016/j.autcon.2018.02.025

44. Pérez-González, C. J., Colebrook, M., Roda-García, J. L., Rosa-Remedios, C. B. (2019). Developing a data analytics platform to support decision making in emergency and security management. Expert Systems with Applications, 120, 167–184. https://doi.org/10.1016/j.eswa.2018.11.023
45. Chen, H. (2018). Evaluation of Personalized Service Level for Library Information Management Based on Fuzzy Analytic Hierarchy Process. Procedia Computer Science, 131, 952–958. https://doi.org/10.1016/j.procs.2018.04.233
46. Chan, H. K., Sun, X., Chung, S.-H. (2019). When should fuzzy analytic hierarchy process be used instead of analytic hierarchy process? Decision Support Systems, 125, 113114. https://doi.org/10.1016/j.dss.2019.113114
47. Osman, A. M. S. (2019). A novel big data analytics framework for smart cities. Future Generation Computer Systems, 91, 620–633. https://doi.org/10.1016/j.future.2018.06.046
48. Nechyporuk, O., Sova, O., Shyshatskyi, A., Kravchenko, S., Nalapko, O., Shknai, O. et al. (2023). Development of a method of complex analysis and multidimensional forecasting of the state of intelligence objects. Eastern-European Journal of Enterprise Technologies, 2 (4 (122)), 31–41. https://doi.org/10.15587/1729-4061.2023.276168
49. Merrikh-Bayat, F. (2015). The runner-root algorithm: A metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature. Applied Soft Computing, 33, 292–303. https://doi.org/10.1016/j.asoc.2015.04.048
50. Poliarush, O., Krepych, S., Spivak, I. (2023). Hybrid approach for data filtering and machine learning inside content management system. Advanced Information Systems, 7 (4), 70–74. https://doi.org/10.20998/2522-9052.2023.4.09
51. Balochian, S., Baloochian, H. (2019). Social mimic optimization algorithm and engineering applications. Expert Systems with Applications, 134, 178–191. https://doi.org/10.1016/j.eswa.2019.05.035
52. Lenord Melvix, J. S. M. (2014). Greedy Politics Optimization: Metaheuristic inspired by political strategies adopted during state assembly elections. 2014 IEEE International Advance Computing Conference (IACC), 1157–1162. https://doi.org/10.1109/iadcc.2014.6779490
53. Moosavian, N., Roodsari, B. K. (2014). Soccer League Competition Algorithm, a New Method for Solving Systems of Nonlinear Equations. International Journal of Intelligence Science, 4 (1), 7–16. https://doi.org/10.4236/ijis.2014.41002
54. Hayyolalam, V., Pourhaji Kazem, A. A. (2020). Black Widow Optimization Algorithm: A novel meta-heuristic approach for solving engineering optimization problems. Engineering Applications of Artificial Intelligence, 87, 103249. https://doi.org/10.1016/j.engappai.2019.103249
55. Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-qaness, M. A. A., Gandomi, A. H. (2021). Aquila Optimizer: A novel meta-heuristic optimization algorithm. Computers & Industrial Engineering, 157, 107250. https://doi.org/10.1016/j.cie.2021.107250

56. Hodlevskyi, M., Burlakov, G. (2023). Information technology of quality improvement planning of process subsets of the spice model. Advanced Information Systems, 7 (4), 52–59. https://doi.org/10.20998/2522-9052.2023.4.06
57. Askari, Q., Younas, I., Saeed, M. (2020). Political Optimizer: A novel socio-inspired meta-heuristic for global optimization. Knowledge-Based Systems, 195, 105709. https://doi.org/10.1016/j.knosys.2020.105709
58. Mohamed, A. W., Hadi, A. A., Mohamed, A. K. (2019). Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. International Journal of Machine Learning and Cybernetics, 11 (7), 1501–1529. https://doi.org/10.1007/s13042-019-01053-x
59. Gödri, I., Kardos, C., Pfeiffer, A., Váncza, J. (2019). Data analytics-based decision support workflow for high-mix low-volume production systems. CIRP Annals, 68 (1), 471–474. https://doi.org/10.1016/j.cirp.2019.04.001
60. Harding, J. L. (2013). Data quality in the integration and analysis of data from multiple sources: some research challenges. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL-2/W1, 59–63. https://doi.org/10.5194/isprsarchives-xl-2-w1-59-2013
61. Orouskhani, M., Orouskhani, Y., Mansouri, M., Teshnehlab, M. (2013). A Novel Cat Swarm Optimization Algorithm for Unconstrained Optimization Problems. International Journal of Information Technology and Computer Science, 5 (11), 32–41. https://doi.org/10.5815/ijitcs.2013.11.04
62. Karaboga, D., Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, 39 (3), 459–471. https://doi.org/10.1007/s10898-007-9149-x
63. Fister, I., Fister, I., Yang, X.-S., Brest, J. (2013). A comprehensive review of firefly algorithms. Swarm and Evolutionary Computation, 13, 34–46. https://doi.org/10.1016/j.swevo.2013.06.001
64. Sova, O., Radzivilov, H., Shyshatskyi, A., Shvets, P., Tkachenko, V., Nevhad, S. et al. (2022). Development of a method to improve the reliability of assessing the condition of the monitoring object in special-purpose information systems. Eastern-European Journal of Enterprise Technologies, 2 (3 (116)), 6–14. https://doi.org/10.15587/1729-4061.2022.254122
65. Khudov, H., Khizhnyak, I., Glukhov, S., Shamrai, N., Pavlii, V. (2024). The method for objects detection on satellite imagery based on the firefly algorithm. Advanced Information Systems, 8 (1), 5–11. https://doi.org/10.20998/2522-9052.2024.1.01
66. Owaid, S. R., Zhuravskyi, Y., Lytvynenko, O., Veretnov, A., Sokolovskyi, D., Plekhova, G. et al. (2024). Development of a method of increasing the efficiency of decision-making in organizational and technical systems. Eastern-European Journal of Enterprise Technologies, 1 (4 (127)), 14–22. https://doi.org/10.15587/1729-4061.2024.298568

67. Tyurin, V., Bieliakov, R., Odarushchenko, E., Yashchenok, V., Shaposhnikova, O., Lyashenko, A. et al. (2023). Development of a solution search method using an improved locust swarm algorithm. Eastern-European Journal of Enterprise Technologies, 5 (4 (125)), 25–33. https://doi.org/10.15587/1729-4061.2023.287316
68. Yakymiak, S., Vdovytskyi, Y., Artabaiev, Y., Degtyareva, L., Vakulenko, Y., Nevhad, S. et al. (2023). Development of the solution search method using the population algorithm of global search optimization. Eastern-European Journal of Enterprise Technologies, 3 (4 (123)), 39–46. https://doi.org/10.15587/1729-4061.2023.281007
69. Mohammed, B. A., Zhuk, O., Vozniak, R., Borysov, I., Petrozhalko, V., Davydov, I. et al. (2023). Improvement of the solution search method based on the cuckoo algorithm. Eastern-European Journal of Enterprise Technologies, 2 (4 (122)), 23–30. https://doi.org/10.15587/1729-4061.2023.277608
70. Raskin, L., Sukhomlyn, L., Sokolov, D., Vlasenko, V. (2023). Multi-criteria evaluation of the multifactor stochastic systems effectiveness. Advanced Information Systems, 7 (2), 63–67. https://doi.org/10.20998/2522-9052.2023.2.09
71. Arora, S., Singh, S. (2018). Butterfly optimization algorithm: a novel approach for global optimization. Soft Computing, 23 (3), 715–734. https://doi.org/10.1007/s00500-018-3102-4