Successful Application of Mivar Expert Systems for MIPRA – Solving Action Planning Problems for Robotic Systems in Real Time

O. O. Varlamov¹, ², ³, D. V. Aladin¹, ²

¹ Bauman Moscow State Technical University, Moscow, Russia
² MIVAR Scientific and Research Institute, Moscow, Russia
³ Moscow Automobile and Road Construction State Technical University, Moscow, Russia

For the first time in the world, mivar expert systems were used for solving action planning problems for robotic systems. The KESMI (Wi! Mi) Razumator solution was used to create a mivar-based system called MIPRA (mivar-based intelligent planner of robot actions), which delivers a qualitative (by several orders of magnitude) boost to the process of solving STRIPS-type problems, i.e., constructing algorithms for planning the actions of a robot tasked with relocating blocks in the blocks-world domain. Practical experiments, which any researcher can verify and replicate, have shown that, instead of many hours and powerful multiprocessor servers, MIPRA runs on an ordinary computer and takes seconds to solve problems involving relocations of the following numbers of blocks: 10 blocks – 0.98 s; 50 blocks – 3.44 s; 100 blocks – 10.57 s; 200 blocks – 38.32 seconds, and 300 blocks – 84.07 seconds. The mivar approach implemented in MIPRA allows it to handle tasks with more than three towers on the table and with varied numbers of blocks, even in changing conditions. This creates an opportunity, in the future, to solve a greater variety of action planning problems for different robots, such as, e.g., ensuring logical compliance with traffic regulations for autonomous vehicles, etc.

Keywords: mivar, mivar networks, decision support system, expert systems, autonomous wheeled vehicles, logical artificial intelligence, driver assistance system, MIPRA, STRIPS, GPS, blocks-world

Introduction

Intelligent planning methods must integrate not only planning methods per se, but also methods for satisfying constraints, temporal reaoning, knowledge representation, formal languages, and models. Also, they present complex technological problems to be solved. Several basic methods of intelligent planning are known, among which are state-space planning, plan-space planning, planning using the forward constraint propagation technique, and precedent-based planning. The first planner to implement state-space planning was STRIPS (STanford Research Institute Problem Solver) [1], which was supposed to be used to solve the problem of forming a behavior plan for a robot tasked with moving objects across many rooms. The idea of the STRIPS algorithm was borrowed from the General Problem Solver (GPS) system [2]. The method used in GPS is called ‘means-ends analysis’ [3]. It implies that, out of all the actions possible in the current state, only those will be considered that are relevant to the goal at hand. STRIPS applies actions without delay, reaching each goal separately.

In 1994, Tom Bylander conducted a study [4] in which he showed that extremely severe restrictions on both the operators and the formulas used in the programming algorithms were required to guarantee polynomial time complexity or even NP-completeness [5] of the planning process. The discovery of new methods and tools for teaching convolutional and recurrent neural networks has stirred the interest of scientists towards exploring the possibility of using deep neural networks for planning tasks [6]. There is also an interest...
in solving problems of intelligent planning using other approaches [7, 8]. Particularly noteworthy are certain works on planning algorithms using a symbolic world model as a basis for acquiring and maintaining knowledge for future use in behavior planning [9–11]. STRIPS is a problem that belongs to the classic blocks-world domain [12], there also being known a version of this problem with four actions [13]. Besides, it is important to consider the possible scenarios where the Sussman anomaly [14] can occur. The analysis of the literature and the results of oral communications with specialists in solving STRIPS programming problems have shown that the currently existing methods remain NP-complete (and even more computationally complex) and do not make it possible, in the general case, i.e. without the use of heuristics, optimizations, and condition simplifications, to solve problems involving as few as 10 blocks in less than 9 hours (with that process being handled by powerful multiprocessor servers and supercomputers). Therefore, we decided to investigate the possibility of using qualitatively different mathematical methods for solving problems of intelligent robot behavior planning including those that belong to the classic blocks-world and STRIPS programming domain.

In 2018, it was proposed to apply mivar expert systems (ES) for solving state-space intelligent robot behavior planning problems and to compare the results with those produced using the STRIPS algorithm [15]. As is known [16], mivar technologies implement the same idea as the GPS method, but also translate it into the formalism of the production approach and Petri nets, thus making it possible to forgo the complete enumeration process and reduce the complexity of logical inference to linear. The metric proposed for the evaluation of autonomy and intelligence of a robotic system (RS) or a cyber-physical system (CPS) was the number of production rules of the ‘if-then’ form describing all possible unit actions in robot behavior planning [17]. This approach allows for the use of mivar expert systems as decision support systems for robots. Mivar technologies are used to solve various tasks in the field of logical artificial intelligence [18], including such tasks as pattern recognition [19], robot behavior planning, compliance with traffic regulations [20], and creating chatbot consultants [21]. A comparative analysis of expert system development environments has shown that the mivar approach holds great promise [22], and mivar ESs themselves are used in the fields of physics [23], expert modeling [24], as well as in combination with simulation systems [25, 26], e.g. for solving road accident analysis problems [27, 28].

It should be noted that the sub-department of the Bauman Moscow State Technical University named “Information Processing and Management Systems” (IU-5) is currently working on the development of mivar technologies within the scope of several research projects focusing on the creation of hybrid intelligent information systems [29] using multi-level rule sets [30], metagraphs [31], anamorphosis methods [32], cognitive computer graphics [33], neural network algorithms [34], temporal [35] databases [36] and their multidimensional descriptions [37]. Such a comprehensive approach [15–37] makes it possible to solve a variety of practical problems in the field of artificial intelligence. In this paper, the authors practically demonstrate the feasibility of using the mivar approach to drastically reduce the computational complexity of solving STRIPS planning problems [15].

It is, thus, evident that the tasks of improving the intelligence of autonomous robots, RS, and CPS have high priority today, and the use of the mivar approach for planning RS actions in real time is reasonable and promising.

**Formalization of Problem of Mivar Planning of Robotic System Actions**

The problem at hand, i.e. planning of block relocation actions of RSs, is a modification from the classic blocks-world domain [12]. The problem considers only blocks, a table, and a robot. The blocks form towers of various heights, some being one block tall. The table can accommodate $M$ ($M \geq 3$) towers. The locations in which towers can be built are numbered from 0 to $M – 1$. The tower numbering coincides with the numbering of the locations in which they are placed (Fig. 1).

Let $N$ be the number of blocks in the planning problem ($N > 0$). Blocks are numbered from 0 to $N – 1$. Let $States$ be the set of all problem states. Let us associate to each $S \in States$ a tuple, $T_s = [T_{0}, T_{1}, \ldots, T_{(M-1)}]$, where the first element corresponds to the set of blocks placed in location no. 0, the second one corresponds to the set of blocks placed in location no. 1, etc. If there is no block

![Figure 1. Blocks-world problem domain](Image)
Let us consider some examples of tuples $T_i$. Fig. 2 shows two domain states, $S1$ and $S2$. $S2$ is derived from $S1$ by means of some set of robot actions on the blocks.

The following tuple corresponds to state $S1$: $[ [3, 0, 1, 2], [], [6, 4, 5] ]$.

The following tuple corresponds to state $S2$: $[ [3, 0, 1], [2], [6, 4, 5] ]$.

The robot has one mechanical end-effector allowing it to relocate blocks. The robot has a machine vision system that provides information about how blocks are arranged in space. With these tools, the robot can perform actions on the blocks, thus transforming one state of the domain into another. Analogously to a version of our problem domain containing four actions [13], let us define our own set of actions available to the robot:

- $MoveCube(i, j)$ – take one block from the top of the tower in location no. $i$ and place it in location no. $j$. If there is another tower in location no. $j$, then put the block in question on top of this tower;
- $MoveCubes(k, i, j)$ – repeat the action $MoveCube(i, j)$ $k$ times;
- $FindTargetCube(i)$ – determine the number of the location occupied by the tower containing the required block no. $i$;
- $CubeOnTarget(i)$ – register in the robot's memory that block no. $i$ has been placed in its target position;
- $WaitNextRequest()$ – wait for further instructions from the planner;
- $Stop()$ – stop the planning process.

Note that the blocks can only be taken from tops of towers and can only be placed on tops of towers or in unoccupied locations. The machine vision system recognizes the positions of the blocks in space and determines the following characteristics for each block:

- $CurrentTower$ – the number of the tower in which the block is located;
- $On$ – the number of the block on top of which the block is located. If the block is located on the table, this parameter is assigned the value ‘–1’;
- $OverFree$ – a checkbox variable showing whether a tower has any blocks on top of the block being considered.

The planning problem at hand is to create an algorithm – an ordered sequence of actions for the robot – to transform the domain’s initial state (the state in which the blocks were before the robot’s manipulations) into the target state (the state into which the blocks-world needs to be brought by means of the robot’s manipulations).

State-Space Planning in Mivar Networks Formalism

Mivar technologies allow for logical processing and automatic construction of algorithms with linear computational complexity. The applicability of mivars to specific problems is restricted by whether the problem can be represented in the formalism of bipartite mivar networks featuring explicit definitions of Objects and Rules of the ‘if-then’ form. In a mivar network, a ‘rule’ can in and of itself represent an algorithm, a service, or a relation within the ‘Thing-Property-Relation’ (TPR) formalism [15–19], the main thing being that a ‘rule’ has an ‘Input’ (a set of input objects) and an ‘Output’ (a set of the rule’s output objects). There is an important limitation for the use of mivar networks: if all input objects receive values, then all output objects must also receive values.
Let us describe an idea of an algorithm that would apply mivar networks to problems such as STRIPS and MIPRA (mivar-based intelligent planner of robot actions). At the first stage, the problem specifications are analyzed in general, and determinations are made of the number of blocks, the restrictions specified for the number of towers, etc. Based on the initial data obtained, a subject-matter-specific mivar network (a subject-matter knowledge base, i.e. a set of production rules and their relations defined in the ‘Objects, Rules’ formalism characteristic of bipartite networks) is constructed. For example, if there are three blocks, then the network will be constructed for the relocation of these three blocks only, and if the problem has 10 blocks in it, then the mivar network will be larger. Next, the ‘Given’ and ‘Find’ statements, i.e. the initial state of the blocks-world and the desired target state, are analyzed. After that, the mivar network starts searching for an algorithm to solve the problem. After the algorithm has been found, it is communicated to the robot’s actuating members to have the required actions executed.

One solution that has been developed to address the blocks-world planning problem at hand is the mivar-based intelligent planner of robot actions (MIPRA).

MIPRA consists of the following modules:

- The planning problem input module is responsible for receiving information about the initial and target states from the external environment;
- The mivar planner is responsible for planning the actions of the robot required to transform the initial state of the blocks-world into the target state.

MIPRA is built on control systems principles: the objects to be controlled are blocks located on the table, and the robot changes the blocks-world state using its end-effector. When modeling objects to be controlled, MIPRA memorizes a set of robot actions. When the virtual objects being controlled reach their target state, the system generates a solution to the current planning task represented as a sequence of robot actions (i.e. an algorithm) that helped it to achieve this state.

MIPRA’s architecture is shown in Fig. 3

Let us describe the mivar planner and its operating principles in more detail. The system receives a planning task formulated as a list describing the initial and final states of the blocks-world. For each problem in the blocks-world domain, our intelligent mivar planner constructs an individual mivar model [16]. The number of objects within the model is determined by the number of blocks in the domain. Such a method of generating a mivar model of the domain allows for a decomposition of the planning problem taking into account expert knowledge and the solution algorithm.

Information about the current state of the domain is fed into the system from the robot’s sensors and machine vision system. With the help of the mivar model of the domain, an action plan is constructed using logical deduction to achieve an intermediate goal. The resulting set of instructions is communicated to the robot for execution. The robot’s actions change the domain state, and the planner again waits for information from the robot’s sensors. The problem-solving process ends when all the blocks have been put in their proper places.

The decision support module was implemented using the KESMI (Wi! Mi) Razumator software product [18].
running on one processor core at a speed of more than 5 million production rules per second. Functionally, Razumator is a logical planner that allows autonomous robots, RSs, and CPSs to independently construct algorithms and solve problems without human interaction. The main features of the MIPRA created for our study is that we rely directly on the planning problem at hand to generate knowledge about the domain and cyclically access the problem’s specifications to achieve the goal.

**MIPRA Operation Algorithm**

To solve the planning problem, it is necessary to make an algorithm for the robot’s actions. The whole algorithm is divided into iterations and steps. During one iteration, the robot should place at least one block in its designated location within the target state under the planning problem. The assembly of each tower begins with the bottom blocks which, in the target state, must sit directly on the table. The blocks that have been placed in their target positions are no longer considered and take no part in the subsequent plan execution process. The algorithm builds towers according to their designated location numbers, from smaller to greater. Please recall that the towers can only be built in specific locations whose numbering and spatial arrangement are stipulated under the planning problem specifications.

In the initial state, blocks rarely sit in their target places. Therefore, to build one tower, the robot needs to take some intermediate actions to find, free, and install blocks. Such intermediate actions are called ‘steps.’ One step can comprise one or more robot actions provided for within the problem domain. It is important to note that MIPRA launches the mivar network to search for an execution algorithm for each step.

The final (required) planning problem solving plan will consist of iterations which, in turn, will be divided into steps. In other words, at each step, the planner will construct subprograms which, at the output, will be combined into an integrated problem-solving program. At the same time, since the number of blocks available for relocation will decrease with each step, the problem dimension will also decrease. Eventually, these steps will come together to form an algorithm returning a complete solution to the problem of how to bring the domain from the initial state to the target state.

It is noteworthy that, at any time, you can change the problem specifications: add or reduce the number of blocks or possible towers. MIPRA will construct a new mivar network taking into account the new specifications and will start the planning process from the current state of your blocks-world.

**Planning Algorithm Implementation Features**

Any planning problem fed into MIPRA is decomposed into subproblems and intermediate goals. The solution algorithm incorporates the idea that the robot should strive, while working on one subproblem, to place at least one block in its target position. The target position of a block is its spatial disposition in the target state to be achieved within the current intermediate goal. If, in the target state, the block should be sitting on the table, then the respective intermediate goal will be described by the following structure:

$$T_i = (\text{Block: } a, \text{Tower: } b, \text{On: } c),$$

where $i$ is the intermediate goal’s number, $i \in (0, N - 1)$; $a$ is the block number within the intermediate goal, $a \in (0, N - 1)$; $b$ is the number of the location in which block no. $a$ should be placed, $b \in (0, M - 1)$; $c$ is the number of the block on top of which block no. $a$ should be placed, $c \in (0, N - 1)$. In this case, the block is placed on the table, therefore $c = -1$.

Associated to each $T_i$ is a checkbox variable, $TF_i$, which takes the value ‘True’ if intermediate goal no. $i$ has been reached. If, in the target state, the block is not sitting on the table, then the intermediate goal will be described by the following structure:

$$T_i = (\text{Block: } a, \text{Tower: } b, \text{On: } c, \text{True: } TF_{i-1}),$$

where $TF_{(i-1)}$ is the checkbox variable of intermediate goal no. $i - 1$ to be achieved before goal no. $i$. Goal no. $i - 1$ must be achieved to proceed to goal no. $i$. This checkbox variable corresponds to the goal associated to block no. $c$.

In this sense, the proposed algorithm runs in a decreasing-length cycle: the first target block is placed on the table first, and then the next blocks are processed, the number of relocatable blocks being reduced at least by one with each iteration. We, thus, get that at each iteration at least one block is placed in the required position, which means that the total number of iterations in the algorithm should not exceed the number of blocks. If some blocks have already been placed in their required positions, then they are no longer relocated (in the general case, the number of blocks that can be placed in their required positions at a certain step of the algorithm tends to be greater than one). Such a connection between $T_i$ and $TF_{(i-1)}$ makes it possible to maintain the correct order of blocks in any tower on the table in the target state. Also, such a division into subproblems prevents any occurrences of the Sussman anomaly [14].

Let us consider an example to clarify the principle of describing an intermediate goal. Let the target state be described by tuple (Fig. 4) $ST = [ [3, 0, 2, 5], [1, 4, 6], [8, 9, 7] ]$.

To make its algorithms easier to explain, MIPRA assigns to its intermediate goals the numbers matching the actual order of blocks in tuple $ST$, i.e. goal no. 0 will be to land block no. 3 in its target position, goal no. 3 will land block no. 5, goal no. 4 will land block no. 1, goal no. 9 will land block no. 7, etc.
Intermediate goal no. 0 will look like this:

\[ T_0 = (\text{Block} : 3, \text{Tower} : 0, \text{On} : -1). \]

Intermediate goal no. 3:

\[ T_3 = (\text{Block} : 5, \text{Tower} : 0, \text{On} : 2, \text{True} : TF_2). \]

Intermediate goal no. 4:

\[ T_4 = (\text{Block} : 1, \text{Tower} : 1, \text{On} : -1). \]

Intermediate goal no. 9:

\[ T_9 = (\text{Block} : 7, \text{Tower} : 2, \text{On} : 9, \text{True} : TF_8). \]

Results of MIPRA Performance Tests

In our study, we used a solution implementing state-space robot action planning that has been specifically developed for this project, the MIPRACubesCore library (version 2.3). This library was implemented in the C# programming language using the.NET Core 2.2 cross-platform framework.

The purpose of our tests was to obtain metrics that characterize the speed and complexity of the planning process. The Razumator component and the MIPRACubesCore library were deployed on a test bench, its components being described in the table below.

MIPRACubesCore tests were performed on planning problems whose domains comprised different numbers of blocks, varying between 5 and 300. Only three towers could be placed on the table. The problems were generated in such a way that the robot would spend as much time as possible relocating blocks (an adverse planning scenario). For a given number of blocks, five different problems were generated. Fig. 5 shows an example of a logical deduction pattern obtained during the set of tests involving 100 blocks, which pattern was used to construct an algorithm for solving the planning problem consisting of 97 rules and going 37 tiers (logical reasoning steps) deep.

In the example involving 100 blocks, a mivar model consisting of 1111 parameters and 400 rules was generated. For comparison: when solving a problem with 300 blocks and three towers, a model with 3311 parameters and 1200 rules is generated. Other characteristics of mivar models are as follows:

- 10 blocks: 121 parameters and 40 rules;
- 50 blocks: 561 parameters and 200 rules;
- 100 blocks: 1111 parameters and 400 rules;
- 200 blocks: 2211 parameters and 800 rules.

The test results demonstrated that the average time taken to solve one planning problem on the above-mentioned test bench is as follows:

- 10 blocks – 0.98 s;
- 50 blocks – 3.44 s;
- 100 blocks – 10.57 s;
- 200 blocks – 38.32 s;
- 300 blocks – 84.07 s.

Table. Test bench for MIPRACubesCore (version 2.3)

| Component                        | Characteristic                        |
|----------------------------------|---------------------------------------|
| CPU                              | AMD Ryzen 32200G, 3600 MHz            |
| Motherboard                      | MSI B350M Pro-VDH (MS-7A38)           |
| RAM                              | 2 x Kingston HyperX HX424C15FB2/8 DDR4–2400 DDR4 SDRAM (Σ = 16077 MB) (dual-channel mode) |
| Video adapter                    | nVidia GeForce GTX1060 MSI PCI-E6144Mb (GTX 1060 ARMOR6G OCV1) |
| Disk drive (for the software)    | Apacer AS350 240 GB (240 GB, SATA-III) |
| Disk drive (for the operating system) | Samsung SSD860 EVO 250 GB (250 GB, SATA-III) |
| Operating system                 | Windows 10 Pro x64 October 2018 Update (version 1809) |
Please note that the number of elementary logical deduction steps required to plan the movements of the robot's end-effector using the classic complete enumeration approach will exceed the factorial of the number of rules. According to our assumption, the computational complexity of the planning problem for blocks will in this case be equal to the “factorial of the factorial of the number of blocks,” since this algorithm is essentially a double enumeration performed over a decreasing-length cycle. It is, thus, clear that, in the case of 300 blocks, this number will exceed 1200! (the total solution time being in excess of hundreds of billions of years).

The use of mivar expert systems for solving the STRIPS problem, which will now be called MIPRA, makes it possible to complete the planning job for 300 blocks in less than 100 seconds, without any optimizations or heuristics. And this is achieved through a new representation of knowledge in the formalism of mivar networks, which helps reduce the computational complexity of the logical deduction process [15–28], including that involved in robot action planning, from factorial to linear and start creating real ‘brains’ for autonomous robots based on ordinary computers. One standalone problem that remains to be addressed is the creation of mivar knowledge bases, but there are other research works devoted to this.

Conclusions and Plans for Further Work

It has, thus, been theoretically justified and proven in practice that the mivar approach and, in particular, logical solving systems [mivar expert systems, e.g. KESMI (Wi! Mi) Razumator (version 2.1)], can and should be used to drastically reduce the computational complexity of action planning tasks for robots, groups of robots, and various multi-level RSs and CPSs, wherever based and whatever their purpose.

The created software product called MIPRA (mivar-based intelligent planner of robot actions), at the heart of which lies the mivar-based Razumator solution, qualitatively reduces the computational complexity and drastically (by several orders of magnitude) speeds up the algorithm construction process for solving robot action planning problems (STRIPS) involving relocations of blocks in the blocks-world domain. Experiments have shown that instead of many hours and powerful multiprocessor servers, our product, MIPRA, runs on an ordinary computer and takes seconds to solve problems (i.e. to construct a planning algorithm for a robot) involving relocations of the following numbers of blocks: 10 blocks – 0.98 s; 50 blocks – 3.44 s; 100 blocks – 10.57 s; 200 blocks – 38.32 seconds, and 300 blocks – 84.07 seconds.

Moreover, MIPRA can handle tasks with more than three towers on the table and with varied numbers of blocks. We emphasize that a qualitative acceleration has been obtained, and this creates an opportunity, in the future, to solve a greater variety of action planning problems for different robots, such as, e.g., ensuring logical compliance with traffic regulations for autonomous vehicles, etc.

In general, at this stage of research, we did not seek to find or justify the optimal algorithm for block relocations, since the main task was to reduce the construction time for at least one algorithm that solves this problem. We believe that optimization is necessary when there are several solutions, but for now we have shown a solution to the find-one-solution-type problem. There are plans, therefore, to do further research which will address the optimization issues as well as the problems of creating mivar knowledge bases.

REFERENCES
1. Fikes R., Nilsson N. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 1971, no. 2, pp. 189–208.
2. Ernst G.W., Newell A. GPS: a case study in generality and problem solving. Academic Press, 1969, 297 p.
3. Simon H. A. The sciences of the artificial. Cambridge, Massachusetts, MIT Press, 1996, 248 p.
4. Bylander T. The Computational Complexity of Propositional STRIPS Planning. Artificial Intelligence, 1994, no. 69 (1-2), pp. 165–204.
5. Gupta N., Nau D. On the Complexity of Blocks-World Planning. Artificial Intelligence, 1992, no. 56 (2–3), pp. 223–254.
6. Ayunts E., Panov A. I. Task Planning in “Block World” with Deep Reinforcement Learning. In: Biologically Inspired Cognitive Architectures (BICA) for Young Scientists. Springer, 2017, pp. 3–9.
7. Svaco M., Jerbic B., Polancem C., Suligoi F. A Reinforcement Learning Based Algorithm for Robot Action Planning. In: Advances in Service and Industrial Robotics, RAAD 2018, Mechanisms and Machine Science, vol 67. Springer, Cham, pp. 493–503.

8. Zhang Yu, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, Subbarao Kambhampati. Plan explicability and predictability for robot task planning. IEEE International Conference on Robotics and Automation (ICRA). Available at: https://arxiv.org/abs/1511.08158 (accessed 15.07.2019).

9. Panov A. I. Behavior planning of intelligent agent with sign world model. Biologically inspired cognitive architectures, 2017, vol. 19, pp. 21–31.

10. Panov A.I., Yakovlev K Behavior and Path Planning for the Coalition of Cognitive Robots in Smart Relocation Tasks. In: Robot Intelligence Technology and Applications 4. Advances in Intelligent Systems and Computing, vol. 447. Springer, Cham, 2017, pp. 3–20.

11. Osipov G. S., Panov A. I., Chudova N. V. Behavior control as a function of consciousness. II. Synthesis of a behavior plan. Journal of Computer and Systems Sciences International, 2015, vol. 54, iss. 6, pp. 882–896.

12. Nils J. Nilsson. Principles of Artificial Intelligence. Tioga, Palo Alto, CA, 1982, 476 p.

13. Estlin T. A. Integrating Explanation-Based and Inductive Learning Techniques to Acquire Search-Control for Planning. Technical Report AI96-250, Department of Computer Sciences, University of Texas, Austin, TX, 1996. Available at: http://www.cs.utexas.edu/~ai-lab/pub-view.php?PubID=51442 (accessed 17.07.2019).

14. Sussman G. J. A Computer Model of Skill Acquisition Elsevier Science. New York, Elsevier Science Inc., 1975, 133 p.

15. Varlamov O. O., Aladin D. V. On the application of mivar networks for intellectual planning of the behavior of robots in the space of conditions. Izvestiya Kabardino-Balkarskogo nauchnogo tsentra RAN, 2018, no. 6-2 (86), pp. 75–82. (In Russian).

16. Varlamov O. O. Evolyutionnyye bazy dannikh i znani diya adaptivnogo sinteza intellektual'nykh sistem. Mivarnoe informatsionnoe prostranstvo [Evolutionary Databases and Knowledge for Adaptive Synthesis of Intelligent Systems, The Mivar Information Space]. Moscow, Radio i svyaz Publ., 2002, 288 p. (In Russian).

17. Varlamov O. O. About metrics of self-sufficiency and intelligence of robotic complexes and cyberphysical systems. Radio industry (Russia), 2018, no. 1, pp. 74–86. (In Russian).

18. Varlamov O. O!Wi!M Expert System Shell as the Novel Tool for Building Knowledge-Based Systems with Linear Computational Complexity. International Review of Automatic Control, 2018, vol. 11, no. 6, pp. 314–325. DOI: https://doi.org/10.15866/ireaco.v11i6.15855.

19. Maksimova A. Yu., Varlamov O. O. Mivarnaya expert system for image recognition based on fuzzy classification and modeling of different subject areas with an automatic extension of context. Izvestiya YuFU. Tekhnicheskie nauki, 2011, no. 12 (125), pp. 77–87. (In Russian).

20. Varlamov O. O., Aladin D. V. About the creation of mivar control systems for monitoring the keeping of traffic rules on the basis of logical kernel (RAZUMATOR) and expert systems. Radio industry (Russia), 2018, no. 2, pp. 25–35. (In Russian).

21. Adamova L. E., Varlamov O. O., Ospov V. G., Chuvikov D. A. About practical implementation of the mivar virtual russian-language text consultant in the banking sector. Izvestiya Kabardino-Balkarskogo nauchnogo tsentra RAN, 2018, no. 6-2 (86), pp. 10–17. (In Russian).

22. Chuvikov D. A., Peterson A. O. Comparative Analysis of Tool Environments for Developing Expert Systems in Different Subject Domains. Promyshlennye ASU i kontrolly, 2016, no. 8, pp. 20–27. (In Russian).

23. Chuvikov D. A., Nazarov K. V. Designing algorithms for solving physics problems on the basis of mivar approach. International Journal of Advanced Studies, 2016, vol. 6, no. 3, pp. 31–50.

24. Chuvikov D. A. Application of expert modeling in new knowledge obtained by man. Radio industry (Russia), 2017, no. 2, pp. 72–80. (In Russian).

25. Chuvikov D. A. Methodology of Combining Expert System and Simulation System. Promyshlennye ASU i kontrolly, 2017, no. 3, pp. 11–18. (In Russian).

26. Chuvikov D. A. The universal algorithm of interaction between expert system and simulation system. T-Comm: Telekommunikatsii i transport, 2017, vol. 11, no. 4, pp. 34–40. (In Russian).

27. Chuvikov D. A. The Use of Expert System “Analysis MVA” and Simulation System VIRTUAL CRASH 3.0 for Solving Tasks of Criminal Expertise Centre. Promyshlennyye ASU i kontrolley, 2017, no. 5, pp. 23–34. (In Russian).

28. Chuvikov D. A. About the expert system “analysis MVA”, based on the concept of the mivar approach. Problemy iskusstvennogo intellekta, 2017, no. 2 (5), pp. 78–88. (In Russian).

29. Chernenkii V. M., Terekhov V. I., Galanukh Yu. E. The structure of the hybrid intelligent information system based on metagraphs. Neirokompyuter: razrabotka, primenenie, 2016, no. 9, pp. 3–13. (In Russian).

30. Samokhvalov E. N., Revunkov G. I., Galanukh Yu. E. Source code generation of software based on multilevel set of rules. Vestnik Moskovskogo gosudarstvennogo tekhnicheskogo universiteta im. N.E. Baumpa. Seriya: Priborostroenie, 2014, no. 5 (98), pp. 77–87. (In Russian).

31. Galanukh Yu. E., Revunkov G. I., Fedorenko Yu. S. Predicate representation of metagraph data model. Informatsionno-ismertelnyye i upravlyaushchye sistemy, 2016, vol. 14, no. 12, pp. 122–131. (In Russian).

32. Terekhov V. I., Chernenkii I. M. Developing an option for decision making using an anamorphic method. Informatsionno-ismertelnyye i upravlyaushchye sistemy, 2016, vol. 14, no. 12, pp. 132–139. (In Russian).

33. Terekhov V. I. Application of Cognitive Computer Graphics for Urgent Data Visualization by Decision-Making Persons. Inzhenernyy zhurnal: nauka i innovatsiy, 2012, no. 3 (3), pp. 25. (In Russian).

34. Terekhov V. I. GPU realization of a neural network face recognition algorithm. Nauka i obrazovanie: nauchnoe izdanie MGTU im. N. E. Baumpa, 2013, no. 12, pp. 367–374. (In Russian).

35. Baldin A. V., Tonoyan S. A., Eliseev D. V. Query language to mivar representation of relational databases, containing information archive from previous human resources systems. Inzhenernyy zhurnal: nauka i innovatsiy, 2013, no. 11 (23), p. 20. (In Russian).
36. Tonoyan S. A., Saraev D. V. Temporal database models and their properties. *Inzhenernyi zhurnal: nauka i innovatsii*, 2014, no. 12 (36), p. 15. (In Russian).
37. Baldin A. V., Tonoyan S. A., Eliseev D. V. Multidimensional description of complex temporal data. *Informatsionno-izmeritelnye i upravlyayushcie sistemy*, 2016, vol. 14, no. 12, pp. 105–110. (In Russian).

**AUTHORS**

*Oleg O. Varlamov*, Dr.Sc. (Engineering), professor, Bauman Moscow State Technical University, Moscow Automobile and Road Construction State Technical University; CEO, MIVAR Scientific and Research Institute, 72, Oktyabrskaya ulitsa, Moscow, 127521, Russia, tel.: +7 (926) 276 76 45, e-mail: ovar@narod.ru.

*Dmitry V. Aladin*, postgraduate student, Bauman Moscow State Technical University; junior research assistant, MIVAR, 72, Oktyabrskaya ulitsa, Moscow, 127521, Russia, tel.: +7 (915) 363 07 79, e-mail: aladin.dv@yandex.ru.

Submitted 14.06.2019; revised 13.07.2019; published online 21.08.2019.