A New Generation Of Rules-Based Approach: Mivar-Based Intelligent Planning Of Robot Actions (MIPRA) And Brains For Autonomous Robots Have Been Created

Oleg Olegovich Varlamov*, Dmitrii Vladimirovich Aladin

Bauman Moscow State Technical University (BMSTU), RI "MIVAR", Moscow, Russia
*Corresponding author. Email: ovarlamov@gmail.com

Abstract

Both hardware and software are needed when creating autonomous robots. There is already huge progress in the equipment field, therefore; robots' software depends on Artificial Intelligence (AI) development. As many scientists note: a lot of robots have been created, however; all of them are without brains. Why are these robots without brains? This research answers this question which is simple and complex at the same time. It also provides a solution for creating logical brains for autonomous robots. Such logical brains have been already created and the robots are being trained now. Famous Terminator robots with mivar logical brains can be created now. However; the robots have been initially taught to “play cubes” millions of times faster than humans. Now it’s possible to train robots to reason logically and plan their complex actions.

A revolution in intellectual robots development has taken place where the scientific world and human societies are now moving towards the new reality of “smart autonomous robots”. In the 21st century "rocket science" is a problem of creating control systems for autonomous intelligent robots. Such robots require sophisticated methods for representing spatial knowledge and controlling exhaustive search in the alternative space. Many scientists have failed to solve problems associated with large representation spaces and to develop search algorithms for traditional robot action planning.

One of such attempts was the STRIPS algorithm (STanford Research Institute Problem Solver) which solves the problem of rearranging *3* pyramids of *10* cubes in *10* hours (*36.000* seconds) even on modern powerful computers. This task which is called Blocks World belongs to the class of NP-complete problems. It is to be recalled that the logical inference problem is also considered NP-complete problem. In the 1990s, creating “logical brains for robots” was quite impossible. Moreover; scientific research throughout the field of AI was frozen i.e. the “second winter of AI". This work is considered as an attempt to melt this winter snow and to breathe a new life into the Rule-based approach. This is done by creating mivar nets with linear computational complexity of inference and planning robots actions. In addition to the well-known “reflex” robot control systems, a logical control level has been added in this work. Moreover; “decision-making systems” for robots, based on mivar expert systems, have been created.

What has been done for science and for humanity in general so far ?

1. It is well-known that the second wave of Artificial Intelligence in the 1980s and the 1990s was based on the" rule-based approach". However; due to the complexity factor of logical inference, it was impossible to get good results. This problem has been solved in this work using a new approach by combining products and Petri nets into "Mivar nets". In these nets, logical inference is performed with linear computational complexity. Now the rules-based approach can

be revived at a new level and truly intelligent expert systems can be created. The expert system shell Wi!Mi product is available now. Various practical tasks have been already created and solved in this research with a processing speed of more than 3 million production rules per second. Thus; a solution for one of the NP-complete problems has been found with linear computational complexity.

2. The global significance of science is as follows: transition from factorial to linear complexity of logical inference allows creating expert systems and decision-making systems for robots at a new level of AI development. These systems work quickly and don't require large computing power. In this work, logical brains for autonomous robots based on mivar expert systems have been created. These brains run on a single processor without heuristics and exhaustive search at a speed of more than 3 million rule's per second. Consequently, constructing intelligent autonomous robots is possible. This leads to widespread of robots application and to a "new wave of AI development".

3. In this research, a specific example of solving the classical STRIPS-planning task of rearranging cubes on three pyramids has been given. In fact, computational complexity of this task is double the factorial. It has been proven by other colleagues that the task of *10* cubes, on powerful servers using STRIPS, can be solved in *10* hours = *36.000* seconds. In this work, a new MIPRA algorithm has been proposed. It is able to solve the *10* cubes task in 1 second. Thus, solution speed has increased *36* thousand times. Generally, in this work, it has been demonstrated that cubes rearranging task complexity has been reduced from "double factorial" NP (N!!) to P (P = 3):(NP=P=3). This achievement allows the entire world scientific community to move towards the intelligent robots creation.

Keywords: AGI, mivar, mivar nets, decision-making system, expert systems, logical inference, autonomous robot, unmanned vehicles, logical artificial intelligence, brains for robots, driver assistance systems, MIPRA, STRIPS, GPS, Blocks World, Knowledge graph.

Summary
To plan their actions, robots can quickly reason based on mivar expert systems with linear complexity of logical inference.

Short name
Mivar Nets Of Logical AI For Autonomous Robotics

1 Introduction

The field of Artificial Intelligence (AI) has attracted a lot of attention in the recent years. Many researches have been conducted on the topic: how AI outplays a person in various games (*1-5*). They claim that the fields of artificial intelligence, cognitive science, and neuroscience have been once again combined based on a common view of the computational foundations of intelligence (*6*). Many AI problems were not solved in the past. Therefore; in this work, there is a suggestion to go back to the 1980s when expert systems (*7*) appeared and various AI capabilities were explored: from programming to robots (*8-10*). During those years and due to the NP-Complete Logical Inference Problem (exhaustive search), it was impossible to solve many practical problems. Therefore; the "winter of AI" started. It should be noticed that the first research on overcoming this limitation, based on mivar nets, appeared in Russia back in 2002 (*11*). Since then there has been huge success in the field of logical AI. This success has allowed talking about the revival of the Rule-based approach.

Artificial Intelligence (AI) techniques are used to create software called "brains for robots". For the first time in the world, Mivar expert systems were used to solve the tasks for planning robot actions. Using RAZUMATOR (MINDer),the constructor of expert systems, Mivar planning system for actions of robots and robotic complexes was created. This system is called MIPRA (Mivar-based Intelligent Planning of Robot Actions). It qualitatively and quantitatively, i.e. several orders of magnitude, speeds up the time to solve problems, such as building a robot

planning algorithm for rearranging cubes in the Blocks World domain. Practical experiments, which can be checked and repeated by any researcher, have shown that instead of many hours of work and powerful multiprocessor servers, MIPRA can solve the problem of rearranging the following number of cubes on an ordinary computer. *10* cubes can be rearranged in *0.46* seconds; *50* cubes – in *3.08* seconds; *100* cubes – in *9.71* seconds; *300* cubes – in *78.01* seconds and *600* cubes – in *330.46* seconds. Mivar approach has been implemented in the proposed MIPRA which works with more than *3* towers on a table and with a variable number of cubes in dynamic conditions. This allows moving on to a greater variety of actions planning on various robots such as the logical observance of traffic laws for autonomous cars, and much more. Mivar expert systems and MIRRA are the basis for creating General Artificial Intelligence (AGI).

Intelligent planning methods besides the actual planning methods often require additional integration with the methods for satisfying constraints, temporal considerations, knowledge representation, formal languages and models, and frequently solving complex technological problems. The main methods of intellectual planning are known. Among these methods are planning in the state space, planning in the space of plans, application of direct propagation of restrictions in planning, and planning based on precedents. The first scheduler to carry out planning in the state space was STRIPS (STanford Research Institute Problem Solver) (*12*). This scheduler was supposed to be used for solving the problem of creating a behavior plan for a robot which moves objects through many rooms. The idea of STRIPS algorithm is borrowed from the GPS system (*13*). The method used in GPS is called means-ends analysis (*14*). It implies considering the current state of only those actions that are relevant to the goal. STRIPS takes action without delay reaching to each goal individually.

In 1994, Tom Bilender conducted a study (*15*) in which he showed that avoiding the polynomial or even the NP-complete (*16*) complexity of planning process requires extremely strict restrictions on both operators and formulas used in algorithms. Discovering new methods and tools for training convolutional and recurrent neural networks aroused scientists' interest to explore the possibility of using deep neural networks in problems planning (*17*). There is also an interest in solving intellectual planning problems using other approaches (*18, 19*). Works in the field of algorithms planning are particularly noteworthy. They use the model of the world sign as basis for acquiring and maintaining knowledge for future use in planning behavior (*20-22*). It is well-known that STRIPS belongs to the classic Blocks World domain (*23*) set of tasks (the world of cubes). A variant of this task with four actions is also known (*24*). In addition, it is important to consider the possibility of Sussman anomaly occurrence (*25*).

As it is known (*26*), Mivar technologies also implement the idea of GPS method. However; they translate this idea into the formalism of production approach and Petri nets which allow switching to linear complexity of the logical inference. As a metric of autonomy, intelligence of robotic systems (RS) and cyber physical systems (CFS), it was proposed to use number of production rules of the “If-Then” format which describe all possible single actions when planning the behavior of robots (*27*). This approach allows the use of Mivar expert systems as decision-making systems for robots. Mivar technologies are used to solve various problems in the field of logical Artificial Intelligence (*16*) including the tasks of driving autonomous cars (*28, 29*), traffic accidents analysis (*30*), and compliance with traffic rules (*31*).

Presently, the Department of “Information Processing and Control Systems” (IU-5) of Bauman Moscow State Technical University is developing Mivar technologies as a part of research work for creating hybrid intelligent information systems (HIIS) (*32*) using unstructured information (*33*), metagraphs (*34, 35*), cognitive computer graphics (*36*), neural networks algorithms (*37*), systems learning algorithms (*38, 39*), intellectual analysis methods (*40*), and classical approaches for developing multilevel systems (*41*). Such an integrated approach (*26-41*) allows solving a wide variety of practical problems in the field of Artificial Intelligence. In this research, it has been practically demonstrated that it is feasible to apply Mivar approach for reducing computational complexity of solving STRIPS tasks planning.

Therefore; tasks of improving autonomous robots intelligence, robotic complexes, and CFS are relevant. Moreover; the application of Mivar approach for planning the actions of robotic systems in real time is justified and it is promising.

2 Results

2.1 Experimental MIPRA performance tests

Logic brains, for autonomous robots, have been created in this research based on mivar expert systems which run on a single processor without heuristics and exhaustive search. In this research, MIPRACubesCore library version 3.1 has been developed. It implements robot actions planning in the space state. This library has been implemented in C # programming language using the cross-platform .NET 5.

The purpose of the conducted tests is to obtain the metrics which characterize speed and complexity of planning process. Razumator (Minder) component and MIPRACubesCore were deployed on a test bench. Composition of this test bench is presented in Table 1.

Table 1. Test bench for MIPRACubesCore version 3.1

Component	Description
CPU	AMD Ryzen 3 2200G, 3600 MHz
Motherboard	MSI B350M Pro-VDH (MS-7A38)
RAM	2 × Kingston HyperX HX424C15FB2/8 DDR4-2400 DDR4 SDRAM (∑ = 16077 МБ) (dual-channel mode)
Hard drive (for the program/executable)	Apacer AS350 240GB (240 GB, SATA-III)
Hard drive (for the operating system)	Samsung SSD 860 EVO 250GB (250 GB, SATA-III)
Operating system	Windows 10 Pro x64 version 2004

MIPRACubesCore testing has been performed on planning tasks where number of cubes in the proposed domain ranges from *5* to *600*. Maximum *3* towers could be placed on the table. Tasks have been generated in such a way that the robot has spent as much time as possible on performing permutations of the cubes (negative planning scenario). For a given number of cubes, *10* different tasks have been generated.

Figure 1 shows an example of a logical conclusion obtained during testing of *100* cubes. In this example, an algorithm for solving planning problem consisting of *97* rules and *37* levels with depth (“37 logical moves”) has been built.

In this example, a Mivar model has been generated. It consists of *1111* parameters and *400* rules. Usually for comparison, when solving a problem with *300* cubes and *3* towers, a model with *3311* parameters and *1200* rules is generated. Other examples can be found in Table 2.

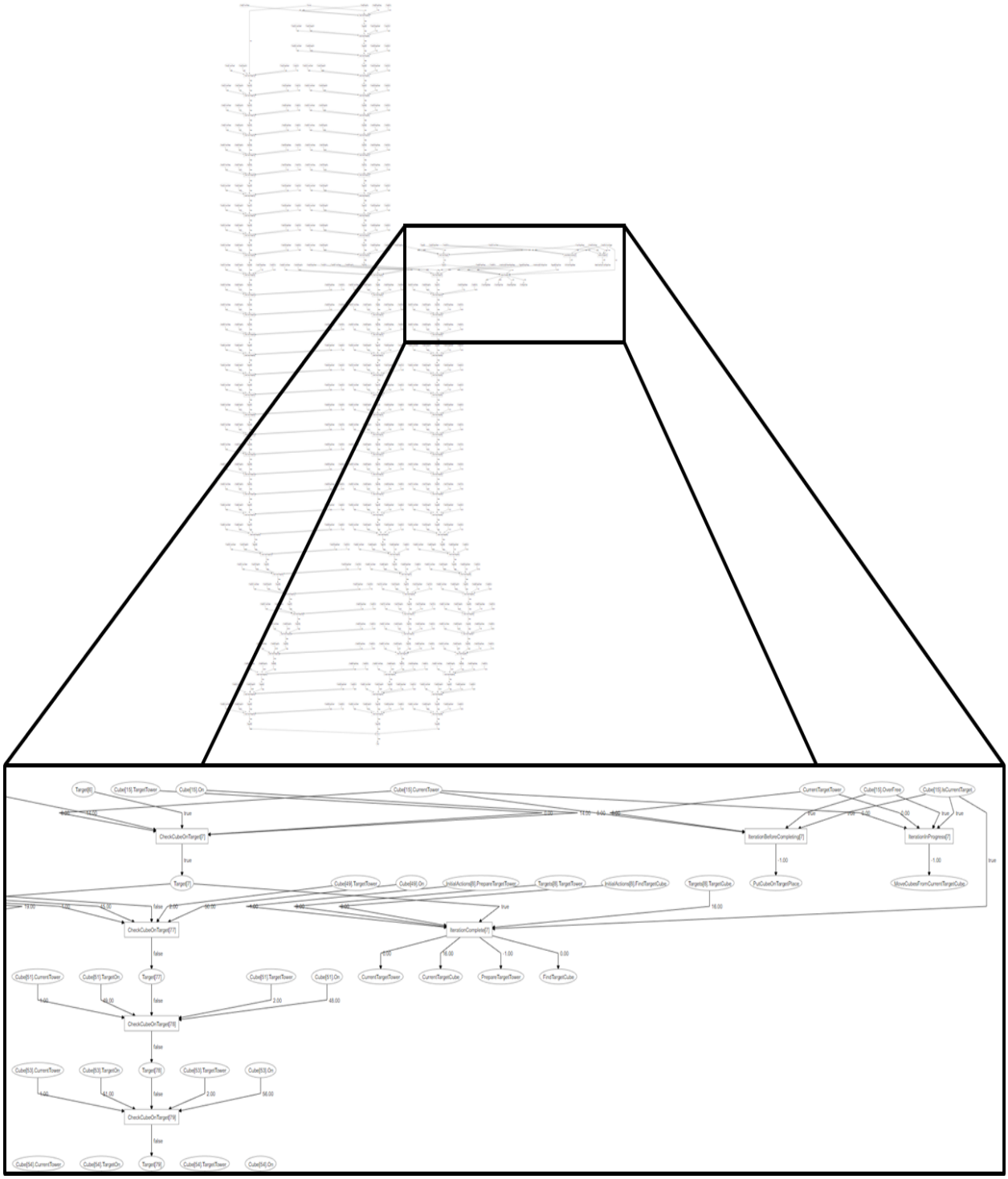


Figure 1. Example on inference for 100 cubes

Table 2. Dependence of number of parameters and rules on number of cubes

Number of cubes	Number of parameters	Number of rules
10	121	40
25	286	100
50	561	200
50	561	200
100	1111	400
150	1661	600
250	2761	1000
300	3311	1200

2.2 Results of the numerical experiments

Total Scheduler Run Time

Time spent to solve the planning problem.

Formulas of the trend line:

$y = 0.0009x^2 - 0.0359x + 2.4013$

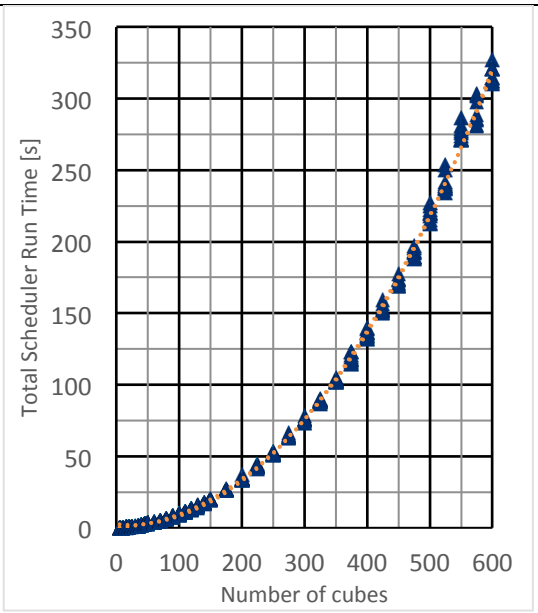
$R^2 = 0.9987$

$y = 3E\text{-}07x^3 + 0.0007x^2 + 0.0225x + 0.1646$

$R^2 = 0.9989$

Calculations according to the trend line formulas:

10 cubes — *0.46* seconds
50 cubes — *3.08* seconds
100 cubes — *9.71* seconds
300 cubes — *78.01* seconds
600 cubes — *330.46* seconds



Model Generation Time

Model generation time is the time for which Mivar model is formed according to the planning task condition.

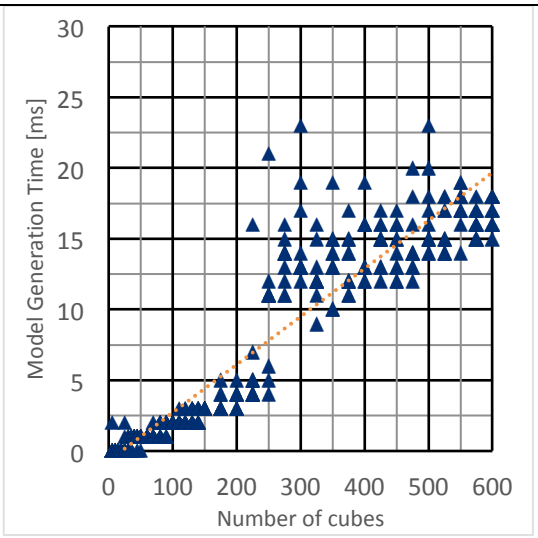
The trend line formula:

$y = 0.0339x - 0.6787$

$R^2 = 0.8804$

Calculations according to trend line formula:

50 cubes — *1.02* milliseconds
100 cubes —*2.71* milliseconds
300 cubes — *9.49* milliseconds
600 cubes — *19.66* milliseconds



Number of parameters

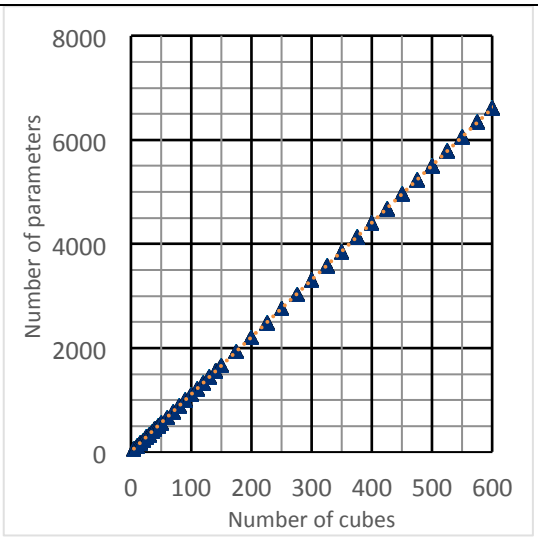
The graph shows how number of the parameters, included in Mivar model, depends on number of the cubes.

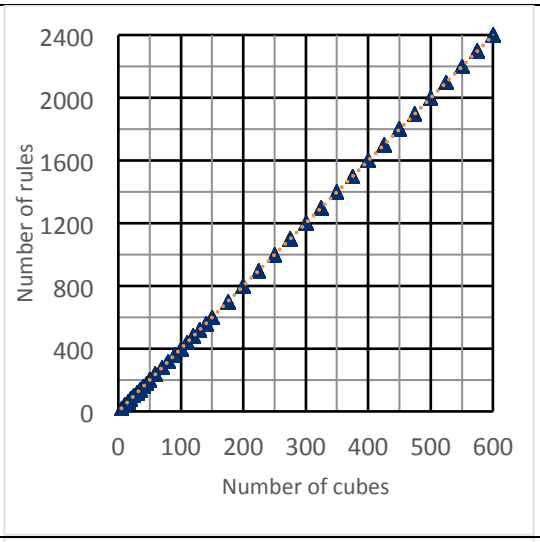
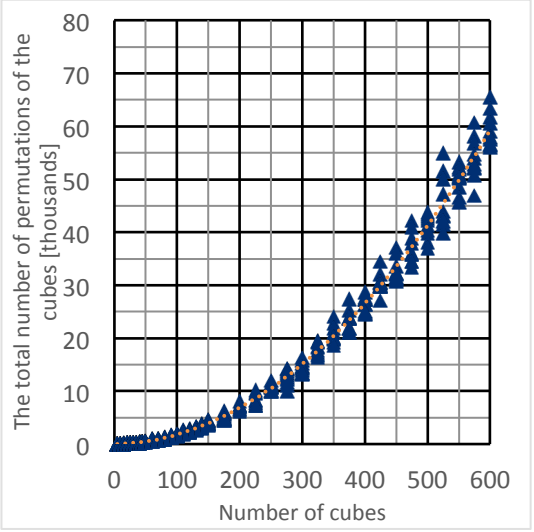
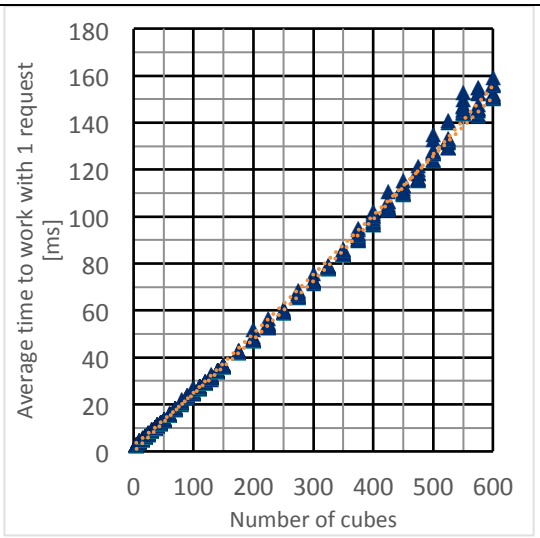
The trend line formula:

$y = 11x + 11$

Calculations according to trend line formula:

10 cubes — *121* parameters
50 cubes — *561* parameters
100 cubes — *1111* parameters
200 cubes — *2211* parameters
300 cubes — *3311* parameters



Number of rules Number of rules that make up Mivar model. The trend line formula: $y = 4x$ Calculations according to trend line formula: *10* cubes — *40* rules *50* cubes — *200* rules *100* cubes — *400* rules *300* cubes — *1200* rules *600* cubes — *2400* rules	
Total number of the cubes permutations Number of the cubes permutations which have been carried out according to the plan. The trend line formula: $y = 0.0002x^2 + 0.0021x - 0.0136$ $R^2 = 0.9926$ Calculations according to trend line formula: *10* cubes — *0.03* thousands of permutations *50* cubes — *0.59* thousands of permutations *100* cubes — *2.20* thousands of permutations *300* cubes — *18.62* thousands of permutations *600* cubes — *73.25* thousands of permutations	
Average working time with 1 request (parsing + processing + generating a response) The average time for which "Razumator" prepares a one-step plan for the robot (one request to "Razumator"). Formulas of the trend line: $y = 7E\text{-}05x^2 + 0{,}2117x + 2{,}6088$ $R^2 = 0.9983$ $y = 0.2519x - 0.1666$ $R^2 = 0.9965$ Calculations based on the trend line formula: *10* cubes — *4.73* milliseconds *50* cubes — *13.37* milliseconds *100* cubes — *24.48* milliseconds *300* cubes — *72.42* milliseconds *600* cubes — *154.83* milliseconds	

2.3 Results of creating mivar brains for the robot which plays with blocks

The experimental results show the following total times required from the scheduler to solve planning tasks:

10 cubes — *0.46* seconds
50 cubes — *3.08* seconds
100 cubes — *9.71* seconds
300 cubes — *78.01* seconds
600 cubes — *330.46* seconds

It is to be noticed that when using the classical approach of exhaustive search, number of elementary actions of logical inference, when planning movements of robot arm-manipulator, exceeds factorial of number of rules. Therefore; it is especially clear that for *300* rules, number of elementary actions is more than *1200!* with a total solution time of more than billions of years.

Using the proposed Mivar expert systems(MIPRA) to solve the STRIPS task, allows planning without any optimization and heuristics for *300* cubes in less than *100* sec. This result is achieved due to a new understanding of knowledge in Mivar net formalism. This understanding allows reducing the computational complexity of logical inference (*26-31*). It also allows reducing the complexity of robots actions planning from factorial to linear and moving on to creating real "brains" for autonomous robots. Mivar expert systems and MIRRA are the basis for creating General Artificial Intelligence.

3 Discussion, conclusions, and plans for further work

In this research, it has been theoretically justified and practically proved that Mivar approach in general and logical decision systems (Mivar expert systems) in particular can and should be applied to reduce the computational complexity of solving the tasks of robots activity planning, as well as their groups, multi-level heterogeneous robotic systems, and cyber-physical systems of various basing and destination.

The created software product "Mivar-based intelligent planner of robot actions"(MIPRA), based on Mivar "Reason", many times (by orders of magnitude) accelerates constructing an algorithm for solving robot actions planning tasks (STRIPS) for rearranging cubes in the Blocks World domain. The experimental results showed that instead of many hours and powerful multiprocessor servers, MIPRA product runs on a regular computer and solves the tasks in seconds. It builds a planning algorithm for the robot by rearranging the following number of cubes: *10* cubes – *0.46* seconds; *50* cubes – *3.08* seconds; *100* cubes – *9.71* seconds; *300* cubes – *78.01* seconds and *600* cubes – *330.46* seconds.

Moreover; MIPRA allows working with more than 3 towers on the table and with a variable number of cubes. It is to be emphasized that a qualitative acceleration has been obtained. This allows going on to a greater variety of action planning for various robots, for example, logical observance of traffic rules for autonomous cars, etc.

The plan for future work is to investigate the balance between number of cubes permutations and total time of the scheduler. Particularly, the plan is to find out the reasons for the polynomial dependence of this parameter on the number of cubes.

In general, at this stage of research, the goal is not to find or substantiate the optimal algorithm for cubes permutation. The main goal is to reduce construction time of at least one algorithm that performs this task. Optimization is necessary when there are several solutions. However; for now, the problem of "finding one solution" has been solved and the solution has been shown. Therefore; all optimization issues will be addressed during further research.

4 Materials and Methods

4.1 Formalizing the task of Mivar planning for robotic complex actions

In this section, actions planning task for the robotic complex to rearrange the cubes must be investigated. This task is a modification of the classic Blocks World domain (*23*) (or the world of cubes). In this problem, only cubes, a table, and a robot are considered. Cubes form towers of

various heights, including one cube high. *M (M≥3)* towers can be placed on the table. Locations of the built towers are numbered from *0* to *M-1*. Towers numbering coincides with numbering the sites on which they are located (Figure 2).

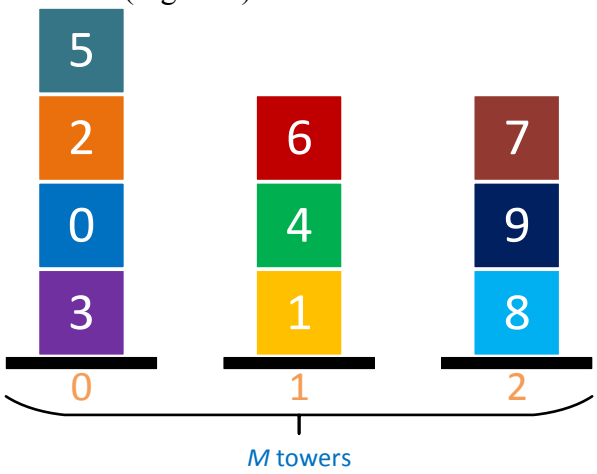


Figure 2. Blocks World task domain.

Let *N* be number of the cubes in the planning problem (*N> 0*). Cubes are numbered from *0* to *N-1*. Let "*States*" be the set of all the states of the problem. Each element *S* of *States* is associated with a tuple $T_s = (T_0, T_1, ..., T_{M-1})$. The first element corresponds to the set of cubes placed on position No. *0* and the second element corresponds to position No. *1, etc*. If there is no stack (or tower) of cubes in place No. *i*, an empty array will correspond to the element *T*, i.e. $T_i = ()$. If some of the cubes are installed on position No. *i*, the element T_i will correspond to the array of cubes numbers ordered from the base to the top of the tower.

Below there are some examples of T_s tuples. Figure 3 shows two states of the domain: *S1* and *S2*. State *S2* is obtained from *S1* by a certain set of actions executed by a robot on the cubes.

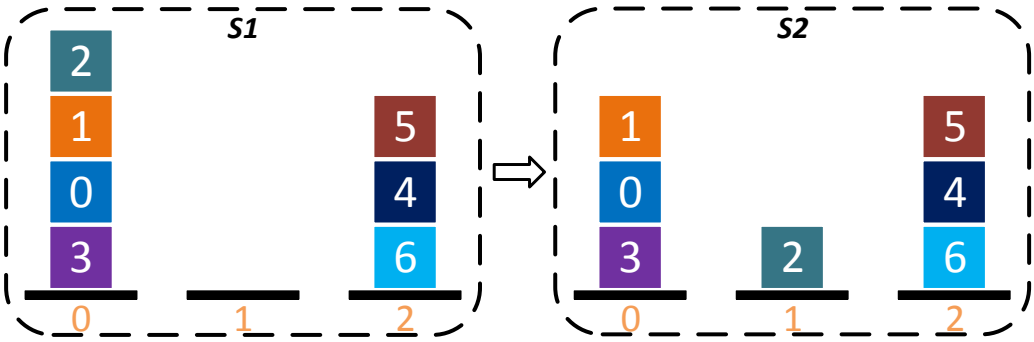


Figure 3. Domain transfer from state *S1* to *S2*.

The state *S1* corresponds to the tuple: ((3,0,1,2), (), (6,4,5)).

The state *S2* corresponds to the tuple: ((3,0,1), (2), (6,4,5)).

The robot has one mechanical manipulator arm which allows it to make permutations of the cubes. The robot has technical vision which provides information on how the cubes are located in the space. Thanks to such tools, the robot can perform actions on the cubes causing transition from one state to another. By analogy with a variant of a task domain with four actions (*24*), a new set of actions available to the robot can be defined as follows.

- *MoveCube (i, j)* performs the following action: take one cube from the top of the tower located on the platform number *i*, and install it on the platform number *j*. If another tower is located on location No. *j*, put the cube in question on the top of this tower.
- *MoveCubes (i, j, k)* repeats the *MoveCube (i, j)* action *k* times.

- *FindTargetCube (i)* determines platform number on which the tower, with the desired cube number *i*, is located.
- *CubeOnTarget (i)* stores in the robot memory that cube number *i* is put in its target place.
- *WaitNextRequest ()* waits for the next instructions from the scheduler.
- *Stop ()* stops planning process.

It should be noticed that the cubes can only be removed from the top of the towers and they can only be placed on the top of the towers or on free sites. Technical vision recognizes position of the cubes in space and determines the following characteristics for each cube.

- *CurrentTower* is number of the tower on which the cube is located.
- *On* is number of the cube on which the described cube is located. If the described cube is located on the table, then -1 is assigned to this parameter as a value.
- *OverFree* is a flag refers to the absence or the presence of a cube in the tower above the described cube.

Denote InitalState is state of the cubes before performing manipulations by the robot (initial state). *TargetState* is the state to which the cubes are to be transferred using robot manipulations (the desired state). Planning task is to create an algorithm, *i.e.* a set of sequential actions by the robot for converting the *InitalState* state to the *TargetState* state (*InitalState* → *TargetState)*.

4.2 Planning in the state space using formalism of Mivar networks

Mivar technologies make it possible to perform logical processing and automatic construction of algorithms with linear computational complexity. However; Mivar method has its limitations. The possibility of using the Mivars depends on the possibility of representing the problem in formalism of Mivar bipartite networks where Objects and Rules of the "If-then" format are explicitly distinguished and separated. A "rule" in Mivar network can itself be a certain algorithm, service, or relation in the "Thing-Property-Relation" (TPR) formalism (*27*). The main idea is that the "rule" has an "Input" which is a set of input objects and "Output" which is represented by a set of output objects. If all input objects receive values, all output objects must receive values as well.

The idea of applying Mivar networks for the STRIPS and MIPRA problems can be described as follows. During the first stage, a general analysis of the problem conditions is carried out. In particular, number of the cubes, the given restrictions on the number of towers, *etc.* are determined during that stage. Based on the obtained initial data, Mivar network of the subject area is constructed (or a knowledge base of the subject area, *i.e.*, the set of production rules and their relationships in formalizing the bipartite net "Objects, Rules"). For example, if 3 cubes are given, the network will be built to rearrange only 3 cubes. If there are 10 cubes in the problem, Mivar network will be larger. Later, the conditions "Given" and "Find" are analyzed, *i.e.* initial state of the cubes and the desired target state. After that, a search for an algorithm to solve the problem starts on Mivar network. After finding this algorithm, it is transmitted to the executive bodies of the robot to perform the necessary actions.

There is another important issue here. It is expected that the robot can monitor state of the source data. In case of data change, a stop occurs and a new search for the solution algorithm is carried out. Mivar network allows to quickly work out such scenarios when number of the cubes changes during course of the task or when it becomes possible to use additional platforms for temporary placement of cubes towers on them.

4.3 Solution Architecture

To solve the planning problem, Blocks World has developed the Mivar-based Intelligent Planning of Robot Actions (MIPRA) which is also called MIPRA in Russian notation (these are two identical abbreviations).

MIPRA consists of the following modules:

- **Planning task input module**. It is responsible of obtaining information from the external environment about tuples of the *InitalState* and the *TargetState* states.

- **Mivar planner.** This module is responsible of scheduling actions of the robot to convert the state of *InitalState* cubes to the *TargetState* state.

MIPRA is built based on the principles of a control system: the control objects are cubes located on the table, and the robot changes the state of the cubes with its manipulator arm. When modeling control objects, MIPRA remembers set of the robot actions. When virtual control objects reach their target state, the system gives, as a response to the planning task, a sequence of robot actions (algorithm) that allows achieving this state. MIPRA Architecture is shown in Figure 4.

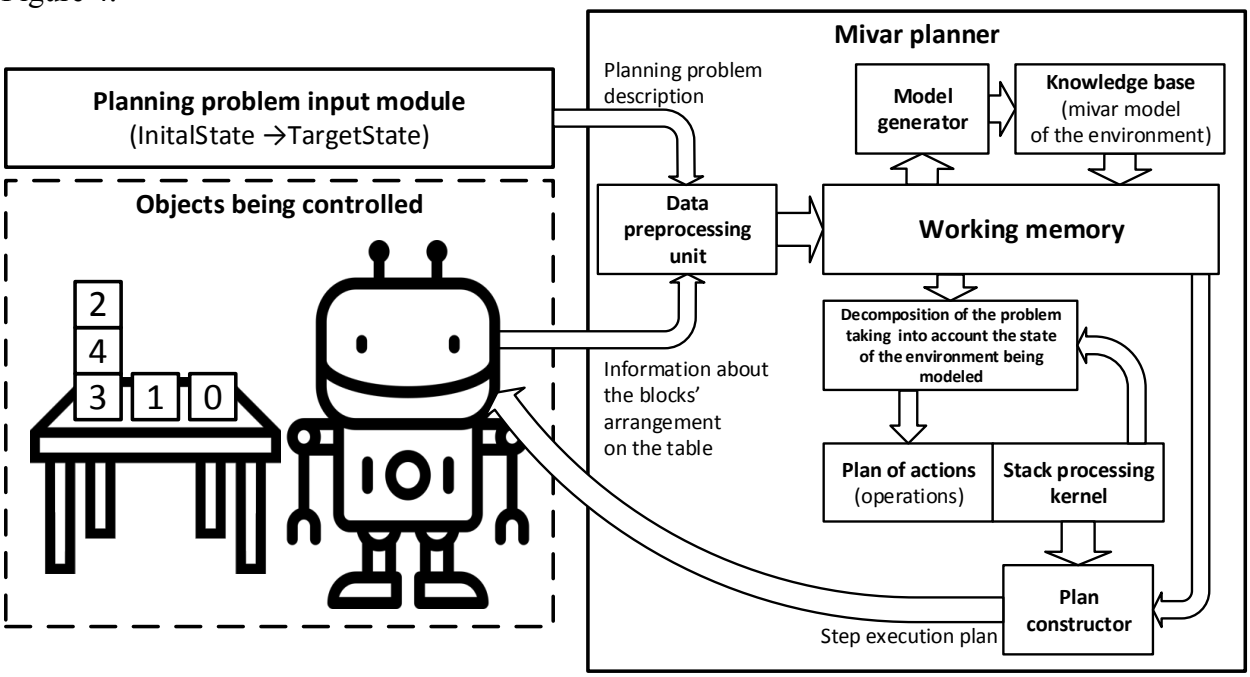


Figure 4. MIPRA Architecture.

5 Mivar planner and its work principles

This module has been developed on basis of Mivar technologies which are as follows (*27*):

1) **Mivar technology of information accumulation** is a way of creating global evolutionary databases and knowledge with adaptively changing structure on the basis of Mivar unified representation "Thing, property, relation".
2) **Mivar information processing technology** is a way to create a logical inference system or "automatically construct algorithms from modules, services or procedures" based on an active learning Mivar net of rules with linear computational complexity. Mivar information processing technology is designed to process information including logical inference, computational procedures, and "services" (based on the development of production approach and services).

The system receives a planning task as a list of the initial and the final states of the cubes. The obtained environmental data are prepared for processing and transferred to the working memory of the system. After that, Mivar domain model is generated. It allows decomposing the planning task taking into account expert knowledge and solution algorithm.

Current state domain information enters the system from the sensors and from the technical vision of the robot. Using Mivar domain model, an action plan is built to achieve an intermediate goal based on a logical conclusion. This set is transferred for execution by the robot. The robot actions change state of the domain, and the scheduler again expects information from the robot sensors. Process of solving the problem ends when all the cubes are in place.

To implement the decision-making module, Wi!Mi "Razumator" ("Minder") software product has been used (*27*). Functionally, "Razumator" is a logical scheduler which enables autonomous robots, RCs, and ~~CFS~~ CFs to independently build algorithms and solve problems without human

participation. Main feature of the created MIPRA in this work is generating information about the subject area directly from the received planning problem, and cyclically referring to it for achieving the goal.

6 MIPRA algorithm

6.1 Principles of forming a robot action plan

In order to solve the planning problem, it is necessary to draw up an algorithm for the robot actions. The whole algorithm is divided into iterations and steps. In one iteration, the robot must at least install one cube in the place provided by the target state of the planning task. Assembly of each tower begins with the lowest cubes which are in the target state on the table. Cubes set on the target places are excluded from consideration and do not participate in further progress of the plan. The algorithm constructs towers in increasing order of sites numbering. It is well-known that the towers can only be built on the sites which have numbering and location provided by condition of the planning task.

Cubes in their initial position rarely stand in their target places. For this reason, to build a single tower, the robot needs to take some intermediate steps to search, release, and install the cube. Such intermediate actions are referred to as step. Therefore; each step may consist of several robot actions provided by the task domain. MIPRA launches a Mivar network to search for the algorithm required for performing each step.

The final (required) plan for solving the planning problem consists of iterations which in turn are divided into steps. In other words, at each step, the scheduler builds a subprogram that is combined at the output into an entire program for solving the problem. At the same time, dimension of the problem decreases at each step because number of the cubes which can be rearranged is reduced. Thus; by sequentially joining the steps, an algorithm for the complete solution of the problem of transition from the initial state to the required state is obtained.

It is important to notice that at any time, conditions of the problem can be changed, *i.e.*, adding or reducing the number of cubes or the possible towers. MIPRA builds a new mivar network taking into account the new conditions and start of the planning process from the current position of the cubes. In the following parts of this work, a detailed description of the scheduler algorithm is considered and the basic terms of MIPRA are explained.

6.2 Target and intermediate states

Any planning task which enters MIPRA is decomposed into sub-tasks and intermediate goals. Solution algorithm depends on the idea that the robot would strive to set at least one cube in its target place for one subtask. Target location of a cube is its location in space when it is in the TargetState domain corresponding to the intermediate target. If the cube in the target state should be placed on the table, the intermediate target is described by the structure of the form:

$$T_i = (Cube: a, Tower: b; On: c),$$

Where:

- i is the intermediate target number; $i \in (0, N-1)$.
- a is the intermediate target cube number; $a \in (0, N-1)$.
- b is site number where the cube number a should be located; $b \in (0, M-1)$.
- c is number of the cube on which cube number a should be located; $c \in (0, N-1)$. In this case, the cube is located on the table, therefore; $c = -1$.

Each T_i is assigned a flag TF_i which takes the value *TRUE* if the intermediate goal No. i is reached. If the cube in the target state is not on the table, the intermediate target is described by the structure of the form:

$$T_i = (Cube: a, Tower: b; On: c, IsTrue: TF_{i-1}),$$

Where:

TF_{i-1} is flag of the intermediate goal number *i-1* which must be achieved before goal number *i*. Achieving the goal number *i-1* is necessary for the transition to achieve goal number *i*. This flag corresponds to the goal of cube number *c*.

In this sense, the proposed algorithm is executed in a an incremental cycle as follows. The first target cube is placed on the table, then the next cubes are processed. In the next iteration, number of the moved cubes is reduced by one at least. At each "iteration", at least one cube is installed in the required place. Total number of "iterations" in the algorithm should not exceed number of the cubes. If some cubes are already installed in the required places, they are no longer rearranged (in general terms, at some step of the algorithm several cubes can be installed in the required places at once). This connection between T_i and TF_{i-1} allows observing the correct order of the cubes in any tower on the table in the *TargetState* state. Moreover; such a division into subtasks ("steps" of the algorithm) ensures that Sussman anomaly doesn't occur (*25*).

An example can clarify the principle of describing an intermediate goal. Figure 5 shows this example where the target state can be described by the tuple ST = ((3,0,2,5), (1,4,6), (8,9,7)).

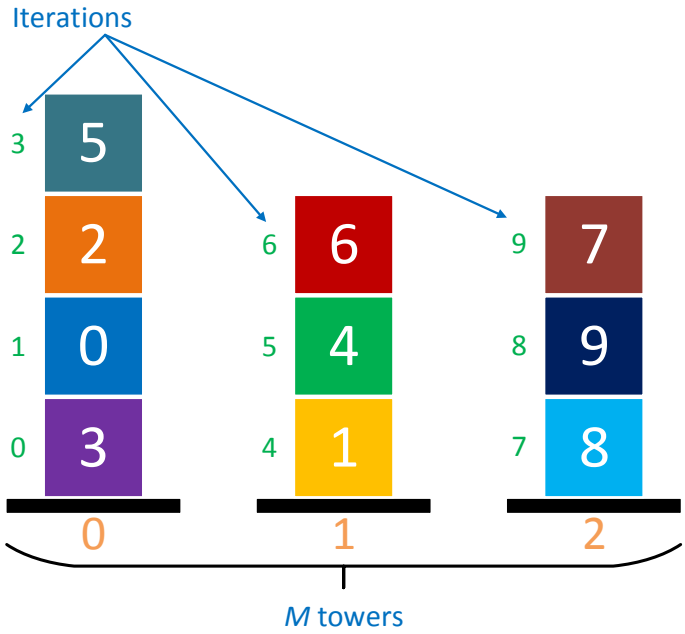


Figure 5. Iterations and towers.

To simplify the algorithm explanation in MIPRA, numbers of the intermediate targets coincide with the actual order of cubes in the *ST* tuple, *i.e.* cube number *3* location at the target location corresponds to goal number *0*, cube number *5* corresponds to goal number *3*, cube number *1* corresponds to goal number *4*, and cube number *7* corresponds to goal number 9, etc.

Intermediate Goal No. *0* is:

$$T_0 = (Cube: 3, Tower: 0; At: -1)$$

Intermediate Goal No. *3*:

$$T_3 = (Cube: 5, Tower: 0; On: 2; IsTrue: TF_2)$$

Intermediate Goal No. *4*:

$$T_4 = (Cube: 1, Tower: 1; At: -1)$$

Intermediate Goal No. *9*:

$$T_9 = (Cube: 7, Tower: 2; On: 9; IsTrue: TF_8)$$

6.3 Step and iteration

A step is a sequential list of robot actions that brings the domain state closer to fulfill conditions of an intermediate goal. Set of steps that allow switching from the domain state of the corresponding target T_i to the target state $T_{(i+1)}$ is called iteration.

Step Example:

```
1. FindTargetCube(1)
2. MoveCubes(0,1,2)
```

During this step, the robot needs to determine position of the cube number *1* and to rearrange *2* cubes from tower number *0* to the platform number *1*.

6.4 Robot action plan

In order to solve the planning problem, it is necessary to consistently achieve all intermediate goals (in the necessary order taking into account the fact that the cubes installed in the required places are no longer rearranged). Cubes placement in the target places begins with the platform number *0* and ends with the platform number *M-1*. The tower is built by collecting the cubes in bottom up manner. In other words, order of achieving intermediate goals coincides with their numbering.

In the tuple ST = ((3,0,2,5), (1,4,6), (8,9,7)) shown in Figure 5, MIPRA draws up a robot action plan in such a way it consistently achieves the goal T_0, T_1, ..., T_9 (cube number *3*, cube number *0*, ..., cube number *7*).

If the numbers a, b, and c are given for the target T_i, then to reach the target T_i, MIPRA makes up a set of steps that together provide the following robot behavior.

1) Number of the site z is required where cube number a is currently located.
2) The number k ($k \notin \{z, b\}$) of the site is determined. Cubes are placed on this site, the thing which prevents achieving the goal T_i. It is important to notice that choosing this number affects the total number of robot actions. The reason is that retracted blocks may interfere with achieving subsequent intermediate goals after T_i. In this research, it has not been decided to achieve efficiency of determining the number k. This part has been left as future work.
3) If necessary, cubes are removed from ~~cube~~ tower No. b (or from the court) in a top-bottom manner where the intermediate goals conditions of these cubes are not fulfilled. Such cubes are transferred to the platform number k.
4) The previous step is performed for cubes located above cube number a.
5) If cube No. a and cube No. c are located at the very top of the towers (or the platform for the cube No. a is free), cube No. a is installed on cube No. c (or on the table).

This behavior is characteristic for all iterations of the plan.

During steps execution, some goals are reached ahead of time. In this case, the planner excludes the cube associated with this goal from consideration and the robot does not do any manipulations with it.

7 Model generation

For each task in the Blocks World subject area, the proposed intelligent planner individually builds a Mivar model. Number of model objects is closely related to number of cubes in the subject area.

Mivar domain description model is composed of two types of objects: classes and parameters (*27*).

A **class** is an abstract entity which generalizes a concept. A class may contain parameters and other classes. The class also has a name, hierarchy level, and description. At least one class must exist in any model. A class with the highest level of hierarchy is called the root class.

A **parameter** is an object that contains a value of a certain type: numeric or text. Like the class, the parameter has a name, hierarchy level, and description.

When creating domain models, the following concepts play an important role:

- **Communication** is association between the objects. It is significant for the subject area. An arbitrary number of input objects through communication is converted into one (or set) of output objects.
- **Relation** is a type of connection, using abstract variables, that describes their interaction.

- A **rule** is a type of connection that binds a relation to specific objects.
- **Restriction** is a type of rule that checks the input data for correctness. For example, side of a triangle cannot be negative.

For any planning task, Mivar model is generated in MIPRA. Class hierarchy of this model satisfies the graph shown in Figure 6. The model is saved in an XML document format file.

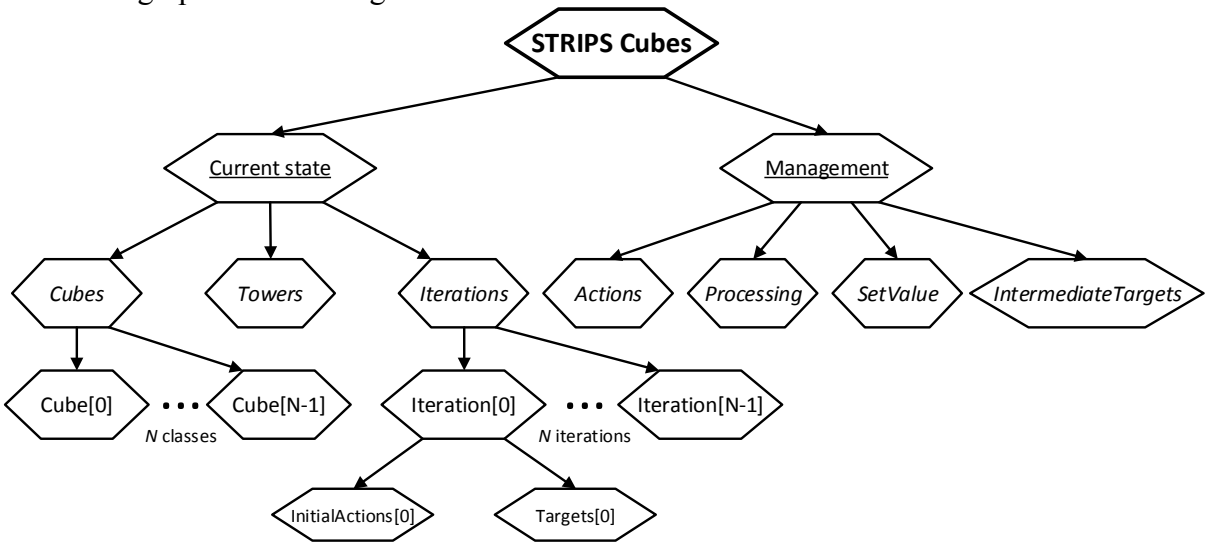


Figure 6. Hierarchy of Mivar classes**.**

The graph in Figure 6 in more detail is as follows.

- Through the "**Current state**" class parameters,"Razumator" (the Minder) receives information about the current and target states of the cubes, the state of the towers, and also about iterations of the plan.
- The "**Cubes**" class describes current state of the cubes.
- Through the class "**Cube (i)**", information about the current position of the cube number *i* in the domain space is received.
- Parameters of the "**Iterations**" class store information about the iteration sequence in the plan. Objects of this class are preliminarily calculated in order to provide a solution to the problem according to the algorithm presented above. They are transferred to the "Razumator" at each request and they participate in formation of a logical conclusion. This solution allows organizing the cyclic process "Scanning the subject area" → "Decision-making" → "Action".
- "**Iteration (i)**" stores information contributing to the achievement of Intermediate Goal No. *i*.
- The class "**InitialActions (i)**" describes available actions of the robot which begin with iteration No. *i*.
- Class "**Targets (i)**" is description of the intermediate target number *i*.
- The "**Towers**" class information stores status of towers.
- Using "**Management**" class, "Razumator" controls the planning process.
- Through parameters of the "**Actions**" class, a list of actions to be performed for the current step is indicated.
- The "**Processing**" class parameters are responsible of managing the planning process.
- Commands of the "**SetValue**" class change the environment variables in the robot view.
- Flags for achieving intermediate goals (or targets) are stored in the "**IntermediateTargets**" class.

Figure 7 shows a screenshot of the model produced by constructor of expert systems called Wi!Mi or "Razumator-Consultant" 2.1 (*27*). The model has been generated to solve a simple problem in the subject area of 3 cubes and 3 towers (which has been done for brevity, because models in practice have been built for 300 cubes, but their display is beyond the scope of this work due to the large volume).

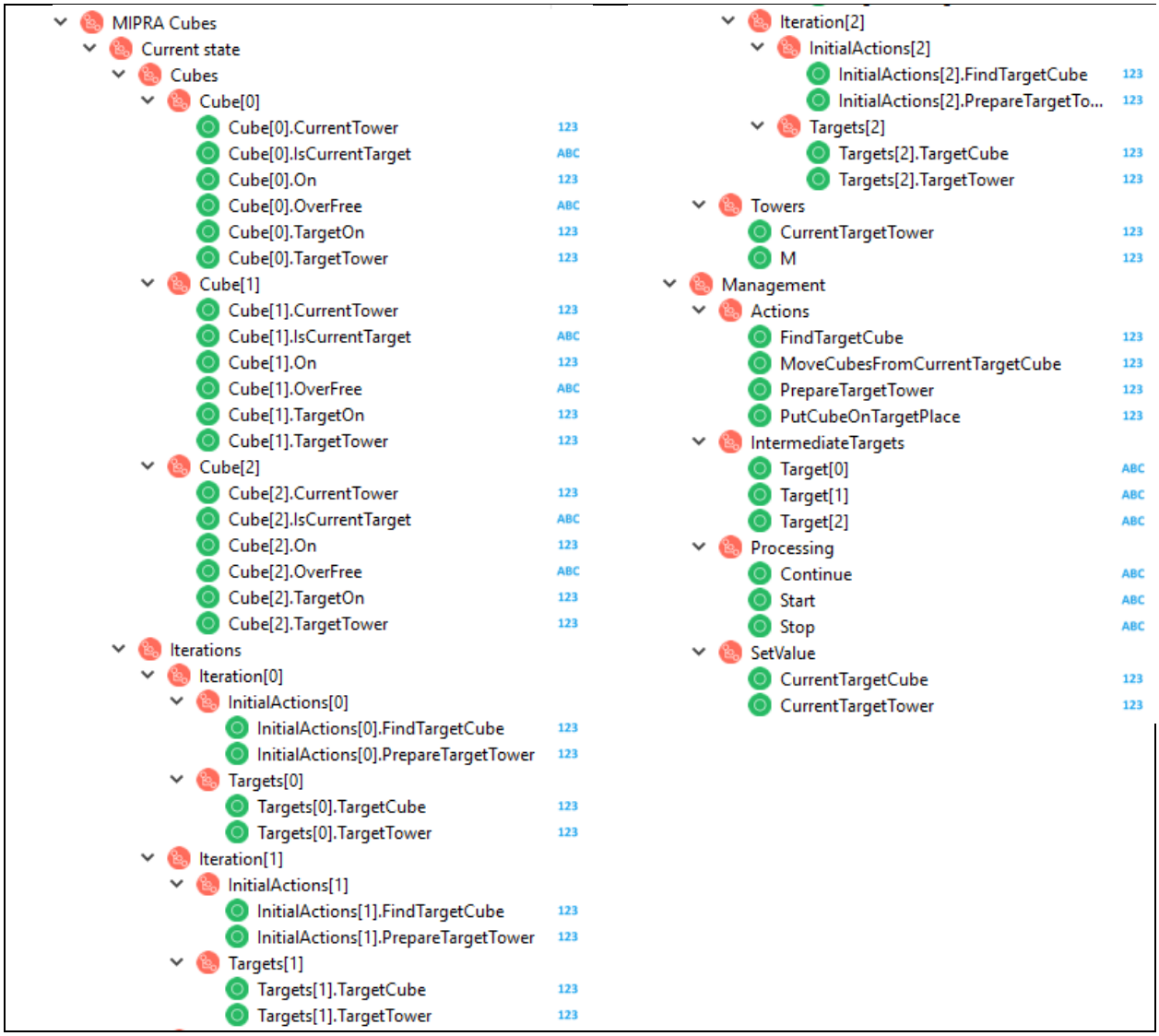


Figure 7. Screenshot of the model from CESMI Wi! Mi "Razumator-Consultant" 2.1

Figure 8 shows an example of an XML document part which contains a class and parameters.

```
class id="d78834c1-54e8-4457-9cde-5f42e8e84b17" shortName="Cube(2)">
  <parameters>
    <parameter id="347bac58-8270-4350-b533-456365db982c"
    shortName="Cube(2).TargetOn" type="double"/>
    <parameter id="50b7380c-344b-4526-8731-897ddf27d218"
    shortName="Cube(2).OverFree" type="string"/>
    <parameter id="80660510-5758-4a2f-950a-b8cc787a9429"
    shortName="Cube(2).IsCurrentTarget" type="string"/>
    <parameter id="b2582d01-8388-4fa3-8806-2932646883fc"
    shortName="Cube(2).TargetTower" type="double"/>
    <parameter id="c77bbfcd-2601-462d-a38b-b47dc487dce7"
    shortName="Cube(2).On" type="double"/>
    <parameter id="ef03de07-5400-46e8-b2b7-591ef702bf6e"
    shortName="Cube(2).CurrentTower" type="double"/>
    </parameters>
    <rules/>
    <constraints/>
    <classes/>
</class>
```

Figure 8. Part of an XML document of a mivar model which contains a class and parameters

The model also has the following relationships:

- **RCheckTarget** checks conditions fulfillment of the intermediate target.
- **RStartWorking** starts planning process.
- **RIterationInProgress** removes objects from the current target cube. The robot is instructed to free the current target cube from interfering objects rearrangement from the top.
- **RIterationBeforeCompleting** puts the current target cube on the target tower if there are no objects interfering from above.
- **RIterationComplete** ends the current iteration and moves on to the next.
- **RFinish** completes planning if the domain is in the target state.

Depending on whether it is required to apply relation for each step or to use it once, rules that link the parameters of mivar model to each other are created. An example of the rules created to solve problem of 3 cubes and 3 towers is presented in Figure 9.

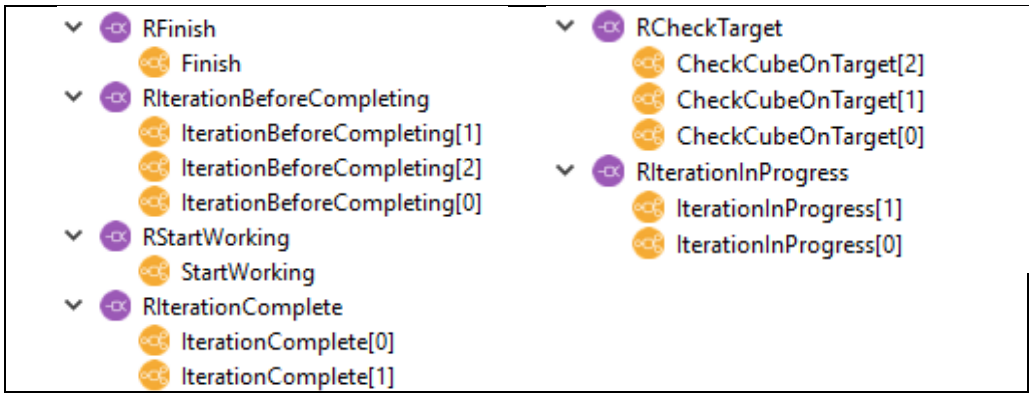


Figure 9. Screenshot of the rules and relationships from constructor of expert systems CESMI Wi! Mi “Razumator-Consultant” 2.1

An example of an XML document containing a relation is shown in Figure 10.

```
<relation     id="6780c60d-8653-4a3e-8a25-76027b2dd5a5"     shortName="RStartWorking"
inObj="Start:string;TargetCube:double;TargetTower:double"         relationType="prog"
outObj="CurrentTargetCube:double;PrepareTargetTower:double;FindTargetCube:double;Curr
entTargetTower:double">
  var Start, TargetCube, TargetTower,
  CurrentTargetCube, CurrentTargetTower,
  PrepareTargetTower, FindTargetCube;
  if ((Start == "true"))
  {
    CurrentTargetCube = TargetCube;
    CurrentTargetTower = TargetTower;
    PrepareTargetTower = 0;
    FindTargetCube = 1;
}
</relation>
```

Figure 10. A fragment of an XML document of Mivar model containing a relation

8 References

1. J. Stajic. Artificial intelligence masters poker. Science 05 May 2017: Vol. 356, Issue 6337, pp. 497. DOI: 10.1126/science.356.6337.497-a.
2. J. Stajic. AI now masters six-player poker. Science 30 Aug 2019: Vol. 365, Issue 6456, pp. 878. DOI: 10.1126/science.365.6456.878-b.
3. N. Brown, T. Sandholm. Superhuman AI for multiplayer poker. Science 30 Aug 2019: Vol. 365, Issue 6456, pp. 885-890. DOI: 10.1126/science.aay2400.
4. Blair, A. Saffidine. AI surpasses humans at six-player poker. Science 30 Aug 2019: Vol. 365, Issue 6456, pp. 864-865. DOI: 10.1126/science.aay7774.
5. N. Brown, T. Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. Science 26 Jan 2018: Vol. 359, Issue 6374, pp. 418-424. DOI: 10.1126/science.aao1733.
6. S. J. Gershman, E.J. Horvitz, J.B. Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. Science 17 Jul 2015: Vol. 349, Issue 6245, pp. 273-278. DOI: 10.1126/science.aac6076.
7. R.O. Duda, E.H. Shortliffe. Expert Systems Research. Science 15 Apr 1983: Vol. 220, Issue 4594, pp. 261-268. DOI: 10.1126/science.6340198.
8. D.G. Bobrow, M.J. Stefik. Perspectives on Artificial Intelligence Programming. Science 28 Feb 1986: Vol. 231, Issue 4741, pp. 951-957. DOI: 10.1126/science.231.4741.951.
9. R.N. Coulson, L.J. Folse, D.K. Loh. Artificial Intelligence and Natural Resource Management. Science 17 Jul 1987: Vol. 237, Issue 4812, pp. 262-267. DOI: 10.1126/science.237.4812.262.
10. R.A. Brooks. New Approaches to Robotics. Science 13 Sep 1991: Vol. 253, Issue 5025, pp. 1227-1232. DOI: 10.1126/science.253.5025.1227.
11. O.O. Varlamov. Evolutionary Databases And Knowledge For Adaptive Synthesis Of Intelligent Systems. Mivar Information Space. (monograph). "Radio and communications", Moscow, 2002. 286p. (in Russian). https://elibrary.ru/item.asp?id=21237254. https://elibrary.ru/download/elibrary_21237254_44481637.pdf
12. R. Fikes, N. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*. **2**, 189-208 (1971).
13. G. W. Ernst, A. Newell, GPS: A Case Study in Generality and Problem Solving (Academic Press, New York, 1969).
14. Simon, H. A. The sciences of the artificial. (Massachusetts: MIT Press, Cambridge, 1981).
15. T. Bylander, The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*. **69**(1-2), 165-204 (1994).
16. N. Gupta, D. Nau, On the Complexity of Blocks-World Planning. *Artificial Intelligence*. **56**(2-3), 223-254 (1992).
17. E. Ayunts, A. I. Panov, Task Planning in “Block World” with Deep Reinforcement Learning. *Biologically Inspired Cognitive Architectures (BICA) for Young Scientists. BICA 2017. Advances in Intelligent Systems and Computing*. **636**, 3-9 (2018).
18. M. Švaco, B. Jerbić, M. Polančec, F. Šuligoj, A Reinforcement Learning Based Algorithm for Robot Action Planning. *Advances in Service and Industrial Robotics. RAAD 2018. Mechanisms and Machine Science*. **67**, 493-503 (2019).
19. Y. Zhang, S. Sreedharan, A. Kulkarni, T. Chakraborti, H. H. Zhuo, S. Kambhampati, Plan explicability and predictability for robot task planning. *ICRA 2017 - IEEE International Conference on Robotics and Automation*. 1313-1320 (2017).
20. A. I. Panov, Behavior planning of intelligent agent with sign world model. *Biologically inspired cognitive architectures*. **19**, 21-31 (2017).

21. A. I. Panov, K. Yakovlev, Behavior and Path Planning for the Coalition of Cognitive Robots in Smart Relocation Tasks. *Robot Intelligence Technology and Applications 4. Advances in Intelligent Systems and Computing*. **447**, 3-20 (2017).
22. G. S. Osipov, A. I. Panov, N. V. Chudova, Behavior control as a function of consciousness. II. Synthesis of a behavior plan. *Journal of Computer and Systems Sciences International*. **54**(6), 882-896 (2015).
23. N. J. Nilsson, Principles of Artificial Intelligence (Tioga, Palo Alto, CA, 1980).
24. T. A. Estlin, Integrating Explanation-Based and Inductive Learning Techniques to Acquire Search-Control for Planning. Tech. Rep. AI96-250, Department of Computer Sciences, University of Texas, Austin, TX, 1996.
25. G. J. Sussman, A Computer Model of Skill Acquisition (American Elsevier, New York. Based on Ph.D. thesis, MIT, Cambridge, MA, 1975).
26. O. O. Varlamov, Exhaustive elementary-incremental summing up of numbers with linear computational complexity. *Avtomatizatsiya i Sovremennye Tekhnologii*. **1**, 34-41 (2003).
27. O. O. Varlamov, Wi!Mi Expert System Shell as the Novel Tool for Building Knowledge-Based Systems with Linear Computational Complexity. *International Review of Automatic Control*. **11**(6), 314-325 (2018).
28. S. S. Shadrin, O. O. Varlamov, A. M. Ivanov, Experimental autonomous road vehicle with logical artificial intelligence. *Journal of Advanced Transportation*. 2492765 (2017).
29. O. O. Varlamov, D. A. Chuvikov, D. V. Aladin, L. E. Adamova, V. G. Osipov, Logical artificial intelligence Mivar technologies for autonomous road vehicles. *IOP Conference Series: Materials Science and Engineering*. **534**(1), 012015 (2019).
30. D. A. Chuvikov, O. O. Varlamov, D. V. Aladin, V. M. Chernenkiy, A. V. Baldin, Mivar models of reconstruction and expertise of emergency events of road accidents. *IOP Conference Series: Materials Science and Engineering*. **534**(1), 012007 (2019).
31. D. V. Aladin, O. O. Varlamov, D. A. Chuvikov, V. M. Chernenkiy, E. A. Smelkova, A. V. Baldin, Logic-based artificial intelligence in systems for monitoring the enforcing traffic regulations. *IOP Conference Series: Materials Science and Engineering*. **534**(1), 012025 (2019).
32. V. Chernenkiy, Y. Gapanyuk, V. Terekhov, G. Revunkov, Y. Kaganov, The hybrid intelligent information system approach as the basis for cognitive architecture. *Procedia Computer Science*. **145**, 143-152 (2018).
33. M. Skvortsova, V. Terekhov, V. Grout, A hybrid intelligent system for risk assessment based on unstructured data. *Proceedings of the 2017 IEEE Russia Section Young Researchers in Electrical and Electronic Engineering Conference, ElConRus*. **7910616**, 560-564 (2017).
34. V. M. Chernenkiy, Y. E. Gapanyuk, G. I. Revunkov, Y. T. Kaganov, Y. S. Fedorenko, Email Author, S. V. Minakova, Using metagraph approach for complex domains description. CEUR Workshop Proceedings. **2022**, 342-349 (2017).
35. V. Chernenkiy, Y. Gapanyuk, G. Revunkov, Y. Kaganov, Y. Fedorenko, Metagraph Approach as a Data Model for Cognitive Architecture. *Advances in Intelligent Systems and Computing*. **848**, 50-55 (2019).
36. V. I. Terekhov, I. M. Chernenky, S. V. Buklin, A. R. Yakubov, Cognitive Visualization in Management Decision Support Problems. *Optical Memory and Neural Networks (Information Optics)*. **28**(1), 27-35 (2019).
37. A. V. Burdakov, A. O. Ukharov, M. P. Myalkin, V. I. Terekhov, Forecasting of influenza-like illness incidence in amur region with neural networks. *Studies in Computational Intelligence*. **799**, 307-314 (2019).
38. A.A. Sukhobokov, Business analytics and AGI in corporate management systems. Procedia Computer Science. **145**, 533-544 (2018).

39. A.A. Sukhobokov, R.Z. Galimov, A.A. Zolotov, A Strategic Management System Based on Systemic Learning Algorithm. *Advances in Intelligent Systems and Computing*. **848**, 290-295 (2019)
40. Y. Gapanyuk, I. Latkin, S. Chernobrovkin, A. Leontiev, G. Ozhegov, A. Opryshko, M. Myalkin, Architecture and implementation of an intelligent news analysis system. *CEUR Workshop Proceedings*. **1975**, 41-55 (2017).
41. A. Ostroukh, N. Surkova, O. Varlamov, V. Chernenky, A. Baldin, Automated process control system of mobile crushing and screening plant. *Journal of Applied Engineering Science*. **16**(3), 343-348 (2018).

9 Acknowledgments

We would like to thank the Bauman Moscow State Technical University (BMSTU) for general financial support. Prof. Oleg Varlamov has developed the general theory of Mivar networks and the corresponding expert systems using bipartite directed graphs. Mr. Dmitrii Aladin has developed the MIPRA algorithm and implemented it as a computer program. Any person who is interested in a practical demonstration of our intelligent planner MIPRA, we can provide him/her access to the demo web application.

We are also grateful to Dr. Valery Karpov for setting the task of comparing mivar expert systems with STRIPS and providing the initial data. Many thanks to Dr. Boris Strokopytov for his invaluable help in preparing and translating the manuscript and Dr. Larisa Adamova for her critical remarks when reading versions of the paper.

10 Supplementary Materials: An example of MIPRA system

10.1 The planning task

In order to demonstrate the operation of MIPRA, the planning task described in Table 3 has been considered

Table 3. The problem statement

Condition	Value
Number of towers	3
Number of cubes	4
Initial state	((2,3),(0),(1))
Target state	((1),(2),(0,3))

Initial state of the cubes in the graphical form is shown in Figure 11.

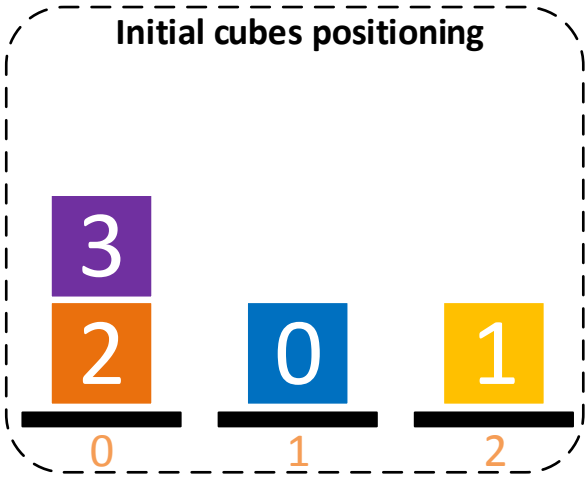


Figure ~~2~~ 11. Initial cubes positioning (initial state).

In order to solve this problem, MIPRA has generated a mivar model which contains *55* parameters and *16* rules. The robot reaches its final goal in *9* permutations of the cube. Demonstration example of the scheduler is simple and is given to illustrate the system principles. Examples of MIPRA work on more complex domains (*3* towers, *10*, *100*, *200* and *300* cubes) can be found on demand as their description is too large for the format of the research. It has been shown in detail how MIPRA can create a plan for a domain of *4* cubes and *3* towers.

10.2 Solution of the problem

At the beginning of each step, the robot examines current location of the cubes on the table using computer vision. Then, it downloads the received information about the situation and auxiliary data for calculating the plan into “Razumator” module which in response gives the necessary set of steps to achieve the intermediate goal. These actions are concretized, taking into account environmental variables of the robot, and they are executed. It has been noticed that steps numbering is continuous for the entire algorithm as well as iterations numbering.

10.2.1 Step No. 0 iteration No. 0

Table 4. The plan for implementing step number 0

Robot actions	Explanation/Comments
MoveCubes(0,1,2) FindTargetCube(1)	The robot sequentially rearranges 2 cubes from tower number 0 to tower number 1. The rearranged cubes did not stand in their places.

	Therefore; they need to be removed from tower number 0. Then the robot searches for position of the cube number 1.

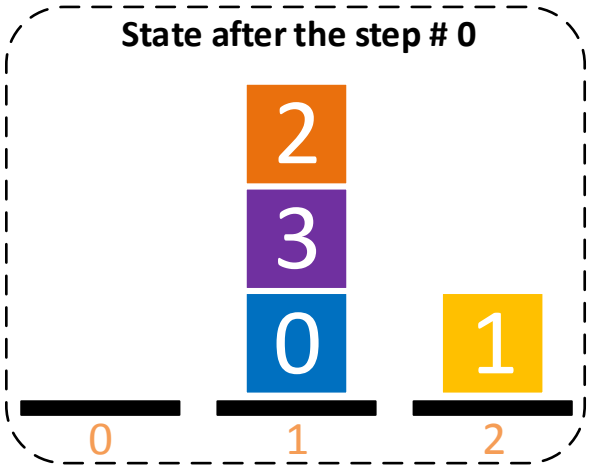


Figure 3. State of the cubes after step number 0.

Table 5. Environmental variables of the robot after step No. 0

Parameter	Value
Current target cube	1
Current tower which contains target cube	2
Current target pad	0
Processed cubes	–
Raw (unprocessed) cubes	{ 0, 1, 2, 3 }
Goals (targets) achieved	–

10.2.2 Step No. 1 of iteration No. 0

Table 6. Execution plan for step No. 1.

Robot actions	Comments
MoveCube (2,0)	Cube number 1 is installed on the platform number 0 (its target location).

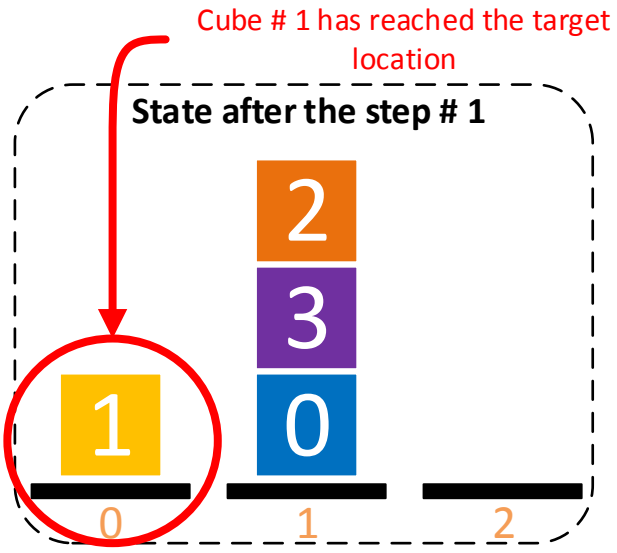


Figure 4. Cubes arrangement after step No. 1

Table 7. Robot environmental variables after step No. 1

Parameter	Value
Current target cube	1
Current tower which contains target cube	0
Current target pad	0
Processed cubes	–
Raw (unprocessed) cubes	{ 0, 1, 2, 3 }
Goals (targets) achieved	–

10.2.3 Step No. 2 of iteration No. 1 (continuous step enumeration for the whole algorithm)

Table 8. Plan for implementing step No. 2

Robot actions	Comments
CubeOnTarget(1) MoveCubes(1,0,3) FindTargetCube(2)	Cube number 1 was at the target location, therefore it is excluded from further consideration. Accordingly, goal No. 0 is considered achieved. The robot sequentially rearranges 3 cubes from platform No. 1 to platform No. 0. The rearranged cubes did not stand in their places, therefore; they need to be removed from platform No. 1. Then the robot searches for position of the cube No. 2.

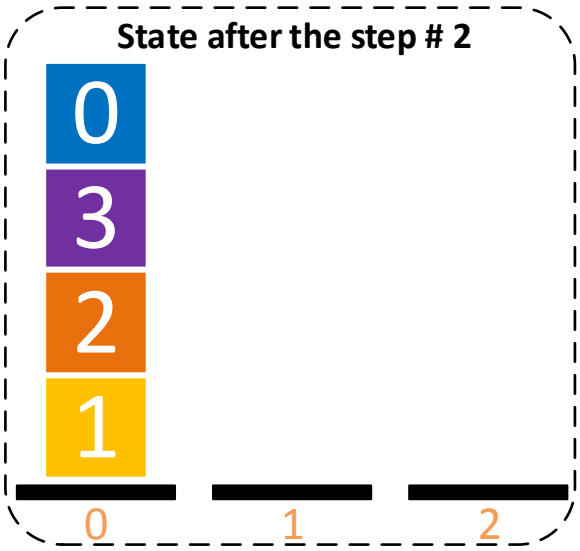


Figure 5. Cubes arrangement after step No. 2

Table 9. Robot environmental variables after step No. 2

Parameter	Value
Current target cube	2
Current tower which contains target cube	0
Current target pad	1
Processed cubes	{ 1 }
Raw (unprocessed) cubes	{ 0, 2, 3 }
Goals (targets) achieved	{ 0 }

10.2.4 Step No. 3 of iteration No. 1

Table 10. Plan for implementing step No. 3

Robot actions	Comments
MoveCubes(0,2,2)	The robot rearranges 2 cubes from tower number 0 to platform number 2. The rearranged cubes interfere with installation of the current target cube in its target location.

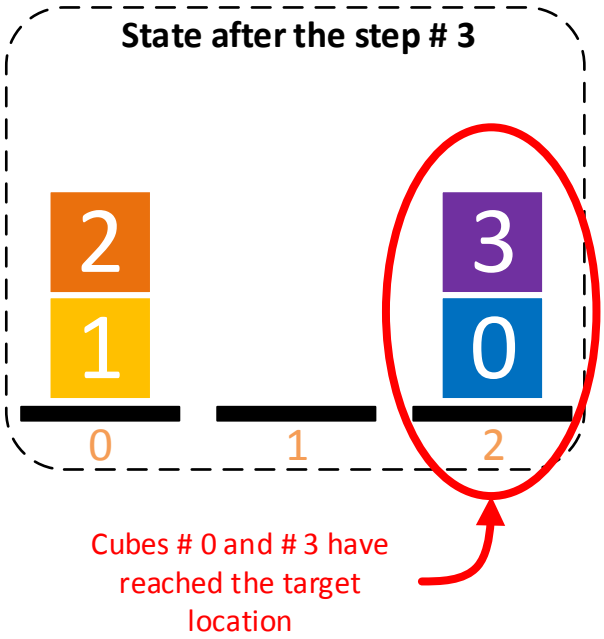


Figure 6. Cubes Arrangement after step No. 3

Table 11. Robot environmental variables after step No. 3.

Parameter	Value
Current target cube	2
Current tower which contains target cube	0
Current target pad	1
Processed cubes	{ 1 }
Raw (unprocessed) cubes	{ 0, 2, 3 }
Goals (targets) achieved	{ 0 }

10.2.5 Step No. 4 of iteration No. 1

Table 12. Plan for implementing step No. 4.

Robot actions	Comments
CubeOnTarget(0) CubeOnTarget(3) MoveCube(0,1)	Cubes No. 0 and No. 3 are at their target positions, therefore; they are excluded from further consideration. Accordingly, goals No. 2 and No. 3 are considered achieved. It is worth noticing that conditions fulfillment of these intermediate goals has occurred outside their iterations. This allows reducing total number of actions that the robot performs during the process of plan execution. Cube number 2 is installed on platform number 1 (its target location).

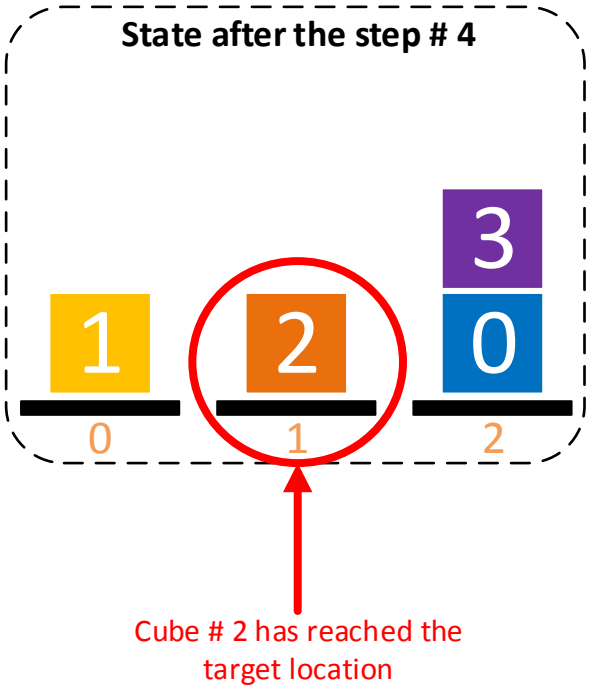


Figure 7. Cubes arrangement after step No.4

Table 13. Robot environmental variables after step No. 4

Parameter	Value
Current target cube	2
Current tower which contains target cube	1
Current target pad	1
Processed cubes	{ 1, 0, 3 }
Raw (unprocessed) cubes	{ 2 }
Goals (targets) achieved	{ 0, 2, 3 }

10.2.6 Step No. 5 of iteration No. 2

Table 14. Plan for implementing step No. 5

Robot actions	Comments
Stop()	All the cubes are at their target places, therefore; Razumator decides to stop the planning process

Cubes arrangement after implementing the last step are shown in Figure 8.

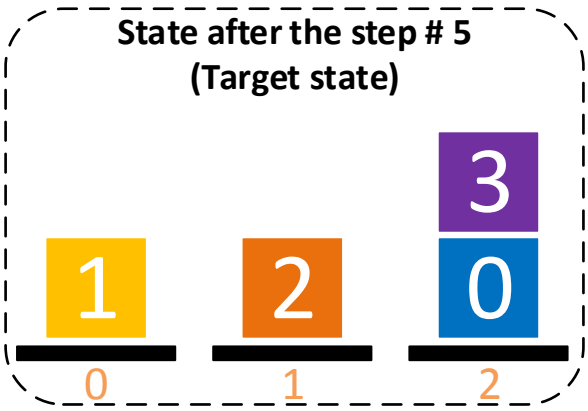


Figure 8. Cubes arrangement after step No.5.

10.3 Logical inference example

Figure 9 shows an example of a logical inference (conclusion). Based on this logical inference, a plan has been adopted for implementing step No. 3.

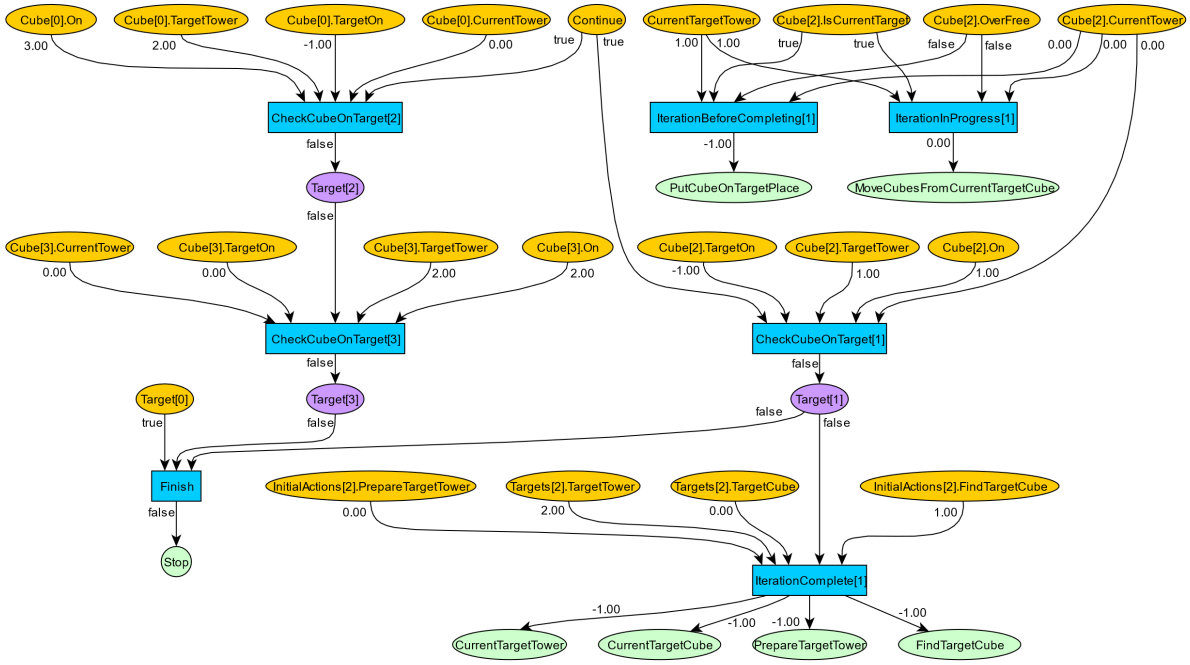


Figure 9. Logical inference example adopted for step No. 3.

At step No. *3*, it is necessary to remove the cubes off cube No. *2*, the thing which interferes with achieving the intermediate goal No. *1* (*Target (1)*). This task is sent to the robot which determines how many cubes to remove from tower number 0 where cube number *2* is located and where to put them.

Mivar model variables are shown in Figure 9 as oval-shaped objects while the logical rules are shown as rectangles. Numerical values that are aligned with the directed edges indicate weights of the edges. If the edge is directed from the parameter, the weight is equal to the value of this variable. In this case, weight of the edge is involved in the rule logic. If the edge is directed from the rule, the weight will be calculated during development of the logical rule. Moreover; it will be assigned the variable value at the end of the edge. Mivar logic can be found in more detail in (*26, 27, 31*).

The variables *Target (1)*, *Target (2)*, *and Target (3)* are calculated in the process of constructing logical inference.

The variables: *Stop*, *CurrentTargetTower*, *CurrentTargetCube*, *PrepareTargetTower*, *FindTargetCube*, *PutCubeOnTargetPlace* and *MoveCubesFromCurrentTargetCube* are the output variables. Rest of the variables are input objects that are sent to the “Sender” from the robot. Input objects represent the current situation on the table, auxiliary variables for organizing the planning process, and information from the robot's memory.

10.4 AI System Model and extract from the research paper (27):

In 2002, a theoretical solution with a linear computational complexity of the logical inference based on Mivar networks was proposed *(11, 27)*. This solution is based on the transition from Petri productions and nets to bipartite oriented Mivar nets which are "figuratively" shown in the Figure 10.

The most important well-known but open question in complexity theory is P versus NP ("P=NP") problem *(11, 27)*. This problem is undoubtedly of great importance for various fields of knowledge. However; this problem could not be solved for more than 40 years. It belongs to the so-called "millennium problems" (officially known as “Millennium Prize Problems”). It is believed that if this problem is resolved, it will be theoretically possible to solve many complex tasks much faster than it is possible now.

There are many problems which are related to the NP-complete class. Logical inference problem belongs to this class of problems.

Transition from NP-complexity of inference search (using the "IF-THEN" productions) to a linear computational complexity of MIVAR method was accomplished and published in 2002 in *(11, 27)*.

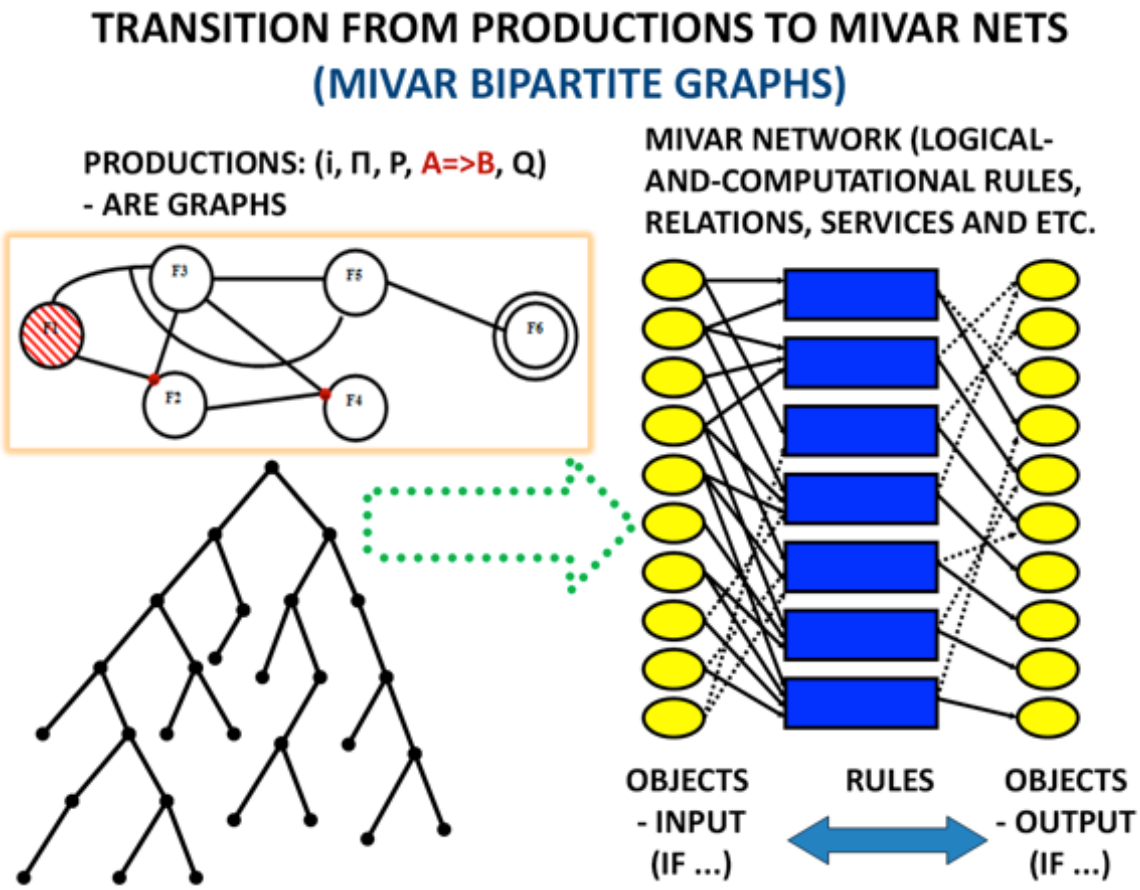


Figure 10. Productions versus MIVAR nets.

Thus, a linear complexity solution was proposed for the NP-complete problem. Hence, since 2002 the logical inference on productions has ceased to be a complete-enumeration task *(11, 27)*. This means only that one of the NP-complete tasks passed to the class of linear problems while all the remaining NP-complete problems remained in this class. The cartoon https://www.youtube.com/watch?v=KxjOYyefJdU shows a qualitative comparison of three generations of ES: a full search, a set of pre-resolved tasks in the database, and MIVAR Razumator (the Reasoner in Russian).

In Russia, studies were conducted on using Mivar nets for various directions in the field of AI which is schematically shown in Figure 11.

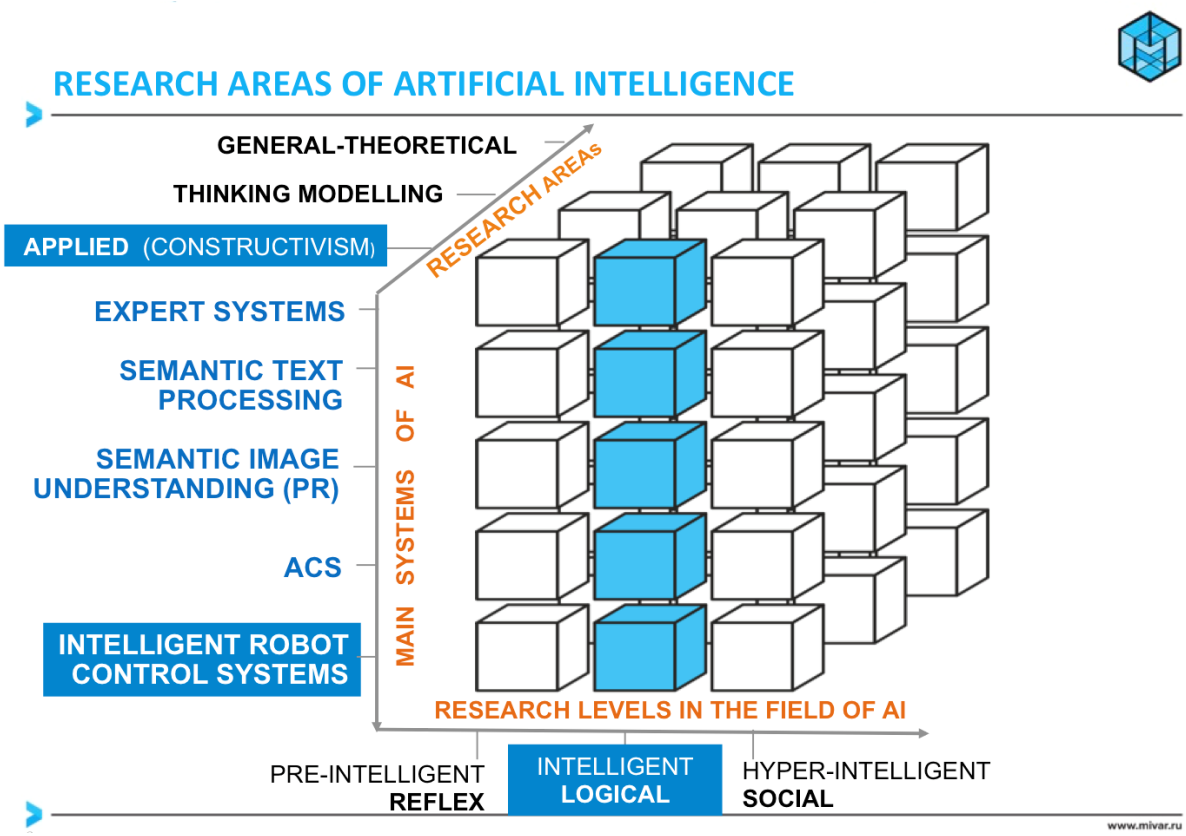


Figure 11. Structural scheme of AI field and the role of MIVAR technology.

For example, it was shown that neural networks can be included in the logical inference path as separate rules based on the principle "input → black box of a neural net → output". There are several useful results of the joint use of Mivar nets and neural nets to understand the meaning of the images published in Russia. However; these applications go far beyond the scope of this work and will be considered in details in the future research. Representatives of the well-known Watson project of IBM had conducted preliminary consultations with us, and some details of the Watson project are known to us. In particular, it is known that this project unites various methods and mechanisms in the field of AI. Total number of those methods exceeds 30 pieces and is constantly growing. There is no direct analogue to the proposed Wi!Mi software in any part of IBM Watson at present time. However; in the long term, the resulted product in this research can be included as a separate cloud service for creating MIVAR knowledge bases and various ESs on subject areas of interest *(11, 27)*. This topic is obviously beyond the scope of this research. However; from a scientific point of view, Mivar nets could well complement IBM Watson product. It should be recalled that the main advantage of MIVAR approach is achieving linear computational complexity of the logical inference on productions. This task was considered NP-complete until 2002 *(11, 27)*, i.e. dramatic reduction in the complexity of solving problems of logical inference and automatic construction of algorithms has been achieved.

The use of both Mivar nets and Wi!Mi Razumator (Minder) has been tested in practice and proved to be efficient in the field of ES (solving "Triangles" problems in real time and ES "Analysis of accidents"). At present, it is used for creating virtual consultants (intelligent chat bots) for Russian banks *(27)*, for the logical situational management of groups of robots, for decision-making in unmanned vehicles *(27)*, and for analyzing traffic rules violations *(27)*. For

example, Figure 12 highlights MIVAR model of ES "Analysis of road accidents" (courtesy of Dmitry Chuvikov).

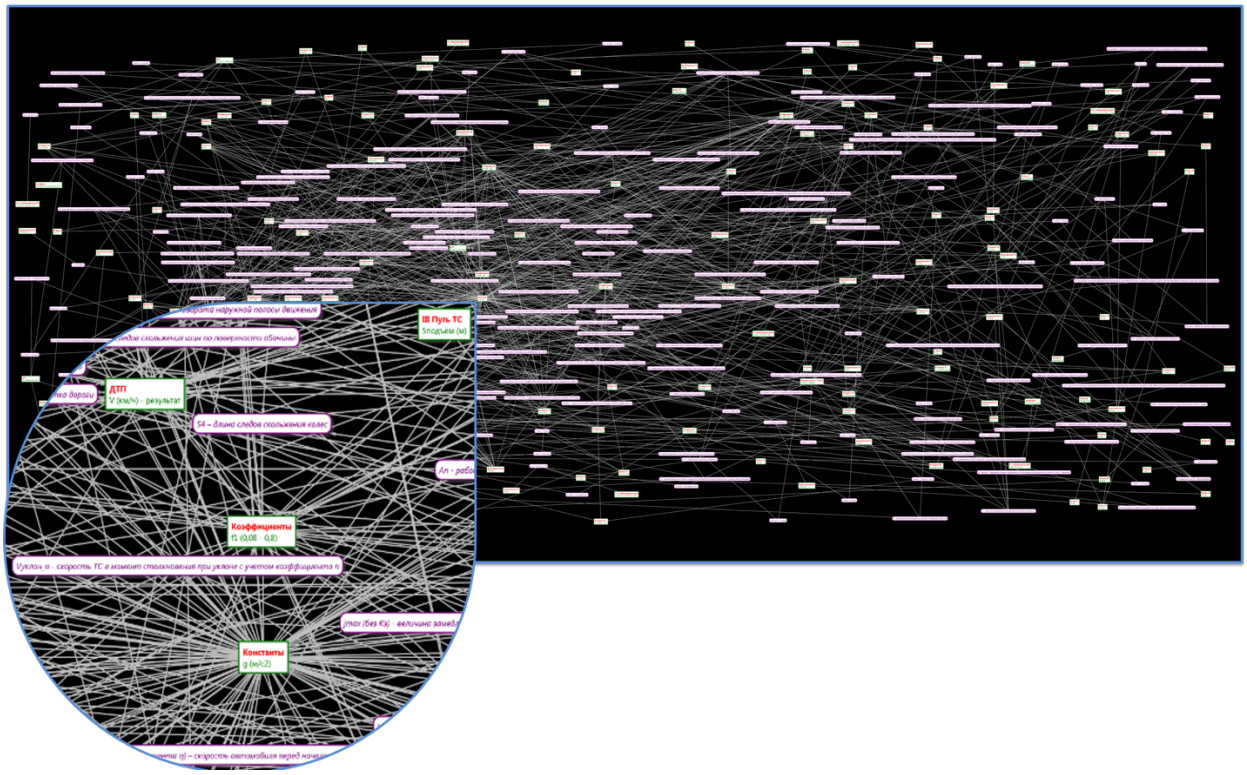


Figure 12. MIVAR model of « road accidents analysis of" subject area. (Created by Chuvikov D.A.:).

10.5 Theoretical basis for a modern expert system shell: Mivar model of knowledge representation.

*O. O. Varlamov, Wi!Mi Expert System Shell as the Novel Tool for Building Knowledge-Based Systems with Linear Computational Complexity. International Review of Automatic Control. **11**(6), 314-325 (2018).*

The most important aspect for knowledge-based systems is the chosen knowledge model representation(Davis, Shrobe, & Solovits, 1993). Knowledge representation is one of the fundamental problems of AI which concerns developers till this day. It can be quoted from (Nebel & Lakemeyer, 2009) that "KR research aims at providing the theoretical foundations on which we can build systems that are useful, comprehensible, and reliable".

ES is generated by borrowing expert knowledge from a person and encoding it into a form which the computer can use to solve similar problems. Therefore; a candidate for a particular ES method of knowledge representation should provide a natural structure for expressing knowledge. Moreover; it should make this knowledge available to a computer and help the developer to describe this knowledge. In addition, this method should allow solving the problems which arise in the subject area. Currently, there are many approaches for knowledge representation such as predicate description, semantic nets, production rules, neural networks, evolutionary, agent-oriented, stochastic, and many others (Russel & Norvig, 2010; Nebel & Lakemeyer, 2009; Luger, 2009). All of these methods are designed to achieve a reasonable compromise between efficiency and representation expressiveness. When creating ES "from scratch" the developer has the opportunity to choose one or another model of representation.

When using ES shells, such choice is often missing, whereas knowledge representation in the shells plays an important role. Wi!Mi shell uses its own MIVAR model of knowledge representation developed by the authors of this research. Basis of MIVAR approach to knowledge representation is adaptive and dynamic description of the modeled domain. The key concept in MIVAR approach to knowledge representation is the notion *mivar net* (MS: MN). This net provides formalization and representation of human knowledge when creating expert systems using Wi!Mi shell.

Mivar net is a way of representing the information part of Mivar space (objects and rules for processing them) as a bipartite directed graph which consists of objects *P* and rules *R* (Figure 7). These objects and rules together form the domain model (Patent No. RUS 2607995, 2015; Varlamov, et al., 2016).

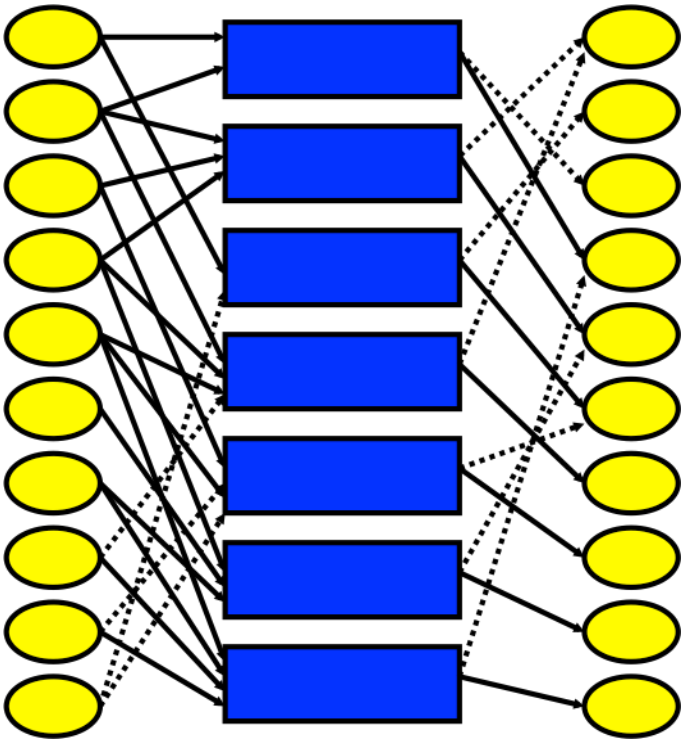


Figure 7. Bipartite graph of mivar net. Objects are represented by yellow ovals and rules are represented by blue rectangles.

Mivar net has the following key features:

1. It consists of two distinct types of elements: objects *(P)* and rules *(R)*;
2. For each variable, information about all the rules *R* for which it is either the input variable *X* or output variable *Y* is clearly indicated.
3. For each rule *R,* information about all its input and output variables *P* is stored including information on the number of input variables *X* and output variables *Y*.
4. All information are stored in database layer specifically adapted to work with mivar nets.
5. For each mivar net item, all related objects and rules are clearly identified. For each mivar node in mivar net, all possible transitions to and from the node are known. Therefore; eliminating the need for exhaustive lookup when searching for inference path in mivar net.

Mivar bipartite graph net can be represented graphically as a two-dimensional matrix of size $M \times N$, (Varlamov, et al., 2016; Varlamov, MIVAR: Transition from Productions to Bipartite Graphs MIVAR Nets and Practical Realization of Automated Constructor of Algorithms, 2017) Where *N* is number of parameters (objects) in subject *area* and *M* is number of rules which link objects of the subject area as clear from Table 1. In each row, all input parameters are marked with a symbol *X,* while all output parameters are marked with symbol *Y*.

Table 1. Mivar net matrix

Parameters / Rules	1	2	3	4	5	...	*N-2*	*N-1*	*N*
1	X	X							Y
2			X		Y			X	X
…									
M		X		X	X		Y	Y	

Various subject areas can be described using mivar nets. As a simple example, the subject areas "Geometry. Triangles" and "Movement of robots" have been considered

Reviewing the domain "Geometry. Triangles" shows the following. Upon entering data from the simplest geometry textbook in the knowledge base, a net of *200* rules and *532* parameters are formed and represented as a *200* × *532* matrix. Part of this mivar net is shown in Table 2.

Table 2. Geometry. Triangles: part of mivar net

Parameters / Rules	Side *AB*	Side *BC*	Side *AC*	Perimeter *P*	Area *T*	…	Altitude *BH*	Median *AM*	Bisection *AL*
Perimeter of the triangle (using lengths of the three sides)	X	X	X	Y					
Area *T* of the triangle using the perimeter and lengths of the sides	X	X	X	X	Y				
…									
Altitude *AH* using the triangle area and length of the side *AC*			X				Y		
Length of the median *AM*		X						Y	

The above described net can be used, for example, as a basis for constructing of the ES-based geometry tutor (Jaquesa, et al., 2013).

Considering the next example of mivar net from subject area “Movement of robots" shows the following. Map of the area on which the robots move is supposed to be known. This area is associated with a table where rows and columns are marked with letters *A, B, C,* ... and so on. A mivar net of movements, as shown in Table 3, can then be created.

Table 3. The simplest example of robots Movements: part of the mivar net

Parameters / Rules	*AA*	*AB*	*BA*	*BB*	*CA*	*AC*	...	*BC*	*CB*
Go to the right (*AA*)	X	Y							
Go forward (*AA*)	X		Y						

Go to the right (*BB*)				**X**				**Y**	
Go forward (*BB*)				**X**					**Y**
Go to the left (*BB*)			**Y**	**X**					
Go backward (*BB*)		**Y**		**X**					
…									
Go to the right (*BA*)			**X**	**Y**					
Go forward (*AB*)		**X**		**Y**					
Go forward (*BA*)			**X**		**Y**				

Thus, mivar net is constructed by connecting sets of two different types according to the two rules: "object-rule" and "rule-object". Relationships like "object-object" and "rule-rule" are prohibited. In general, relationship has the following form: "object(s)-rule-object(s)". The first element specified is an element *from* which the corresponding relationship comes. The second element is an element *where* the relationship goes. Due to this procedure, the graph becomes oriented. This eliminates the possibility of misinterpretation or incorrect transformation of the objects because of the occasional backtrack passage on the relationship. ES of this structure is scalable since it is possible at any point in time to add new sets of elements of any type without having to change their processing methods. In addition, in order to describe such mivar net in many cases, it is simply enough to type the currently existing objects and relationships (rules).

10.6 Mivar inference method

*O. O. Varlamov, Wi!Mi Expert System Shell as the Novel Tool for Building Knowledge-Based Systems with Linear Computational Complexity. International Review of Automatic Control. **11**(6), 314-325 (2018).*

The above described Mivar knowledge representation gives a possibility to construct an algorithm (method) that allows building solution scheme of different problems which occurr in considered subject domains. Using the above presentation of domain knowledge, it is possible to construct an algorithm that allows searching for information within mivar net. Mivar net data model allows determining the open and hidden relationships between objects within the net. Moreover; it allows building algorithms for calculating inference path for any given task within the relevant subject area.

The proposed method is a corner stone of the inference mechanism used in Wi!Mi software environment. Essence of the method is that, for a net of rules presented in a list form, the matrix is built. By analyzing the matrix, inference path can be identified or this path might not exist. In cases when several possible paths exist, the best shortest path is chosen on basis of some optimality criteria.

Dwelling deeper in this method shows the following. If there is a subject area which can be described by *M* rules and *N* variables, Mivar net for this domain can be described by the matrix shown in Table 4.

Table 4. Mivar net of an abstract domain

Parameters / **Rules**	1	2	3	4	5	...	*N-2*	*N-1*	*N*
1	**X**	**X**	**X**					**Y**	**Y**
2			**X**	**Y**	**Y**			**X**	**X**
…									
M		**X**		**X**	**X**		**Y**		

The process of inference path formation can be illustrated using bipartite graph form as clear from Figure 8.

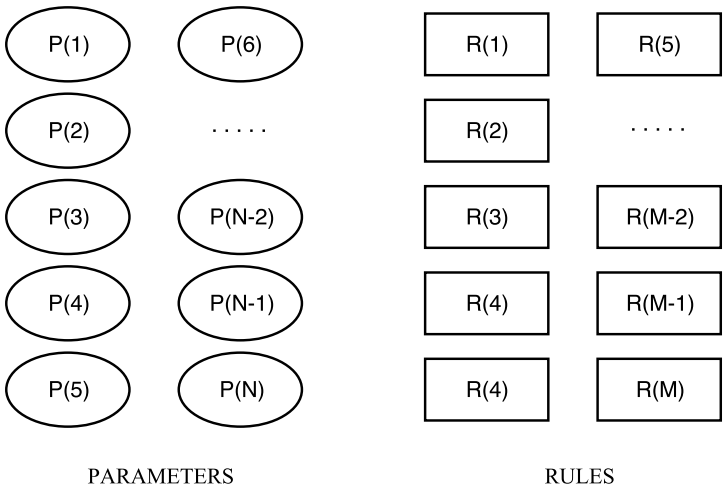


Figure 8. Bipartite graph of an abstract domain

This matrix represents relationships between objects in the given domain. Every row of the matrix corresponds to one of the subject area rules and contains information about variables used in the rule. As before, all input parameters are marked with *X* symbol and all output parameters are marked with *Y* symbol. The following example can be considered. If values of parameters

"*1*", "*2*", "*3*" are given, it is necessary to find the value of the "*N - 2*" parameter. In this case, additional service row with index "*M + 1*" and a column with index "*N + 1*" are introduced. The service row is designed to track changes in the known data. The service column is designed to track usage of the rules. Further actions being performed on mivar matrix are as follows.

1. In the *M + 1* service line, the known parameters are marked as *Z,* and the ones that need to be computed are marked as *W*. For the example illustrated in Table 4 and Figure 8, line *M + 1* symbol *Z* marks elements "*1*", "*2*", "*3*" and symbol *W* marks element "*N-2*" (Table 5, Figure 9).

Table 5. The mivar net domain after the first stage

Parameters / **Rules**	1	2	3	4	5	...	*N-2*	*N-1*	*N*	*Service column*
1	X	X	X					Y	Y	
2			X	Y	Y			X	X	
…										
M		X		X	X		Y			
Service row	Z	Z	Z				W			

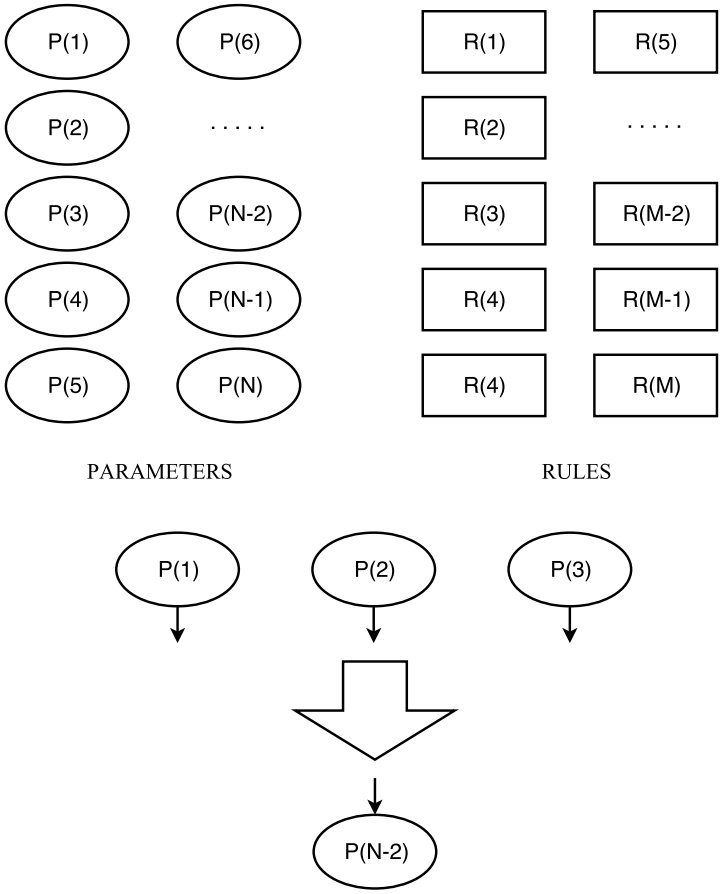


Figure 9. Bipartite graph of inference process: first stage

2. A search can be performed for rules that can be activated, i.e. finding rules for which all of the input variables are known. If there are no more rules which can be triggered, the inference path (the algorithm for solving the problem) does not exist. In this case, clarification (adding) input variables or rules is required. If such rules exist and can be activated, it is marked so in the service column. For example, the sign " ✔ " can be added into the matrix in these positions as shown in Table 6 in cell with indices *(1, N + 1)*. The result is also shown in the form of bipartite graph (Figure 10).

Table 6. Mivar net of the subject domain after the second stage

Parameters / **Rules**	1	2	3	4	5	...	*N-2*	*N-1*	*N*	*Service column*
1	X	X	X					Y	Y	✓
2			X	Y	Y			X	X	
…										
M		X		X	X		Y			
Service row	Z	Z	Z				W			

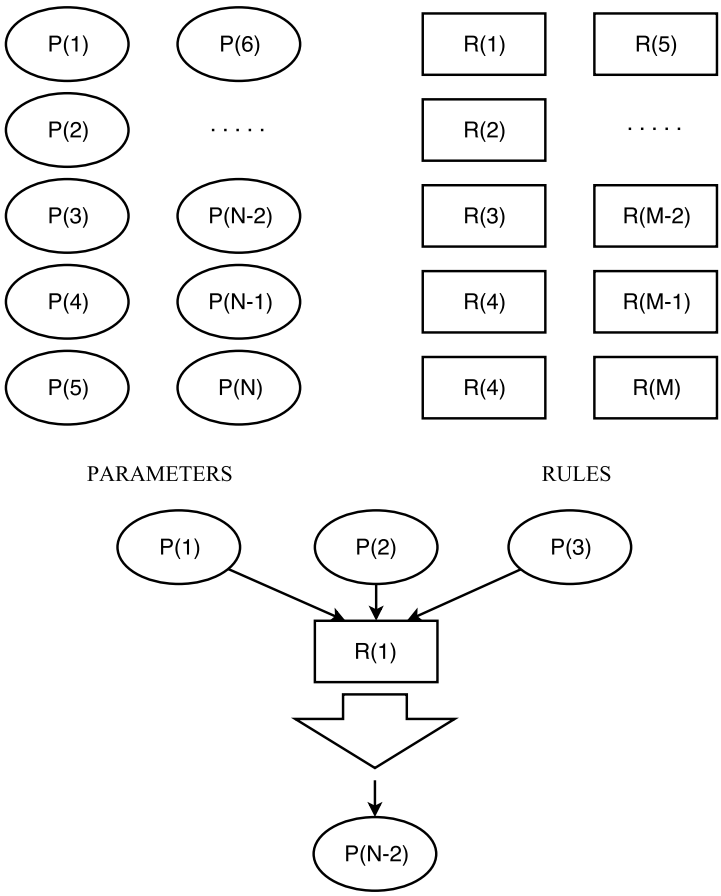


Figure 10. Bipartite graph of inference process: second stage

If several rules can be activated, selection is carried out according to pre-defined criteria to decide which of them must be activated first. With sufficient resources, multiple rules can be simultaneously run.

3. A rule start simulation is done by marking the "known" output variables with the *Z value*. The activated rule, for convenience, can also be marked using the sign "✓✓" but this is not mandatory.

4. After simulation, rule processing output needs to be analyzed to check whether the final result has been computed. If the service line $M + 1$ contains at least one desired undefined value (W), an algorithm should continue further search for inference path. Otherwise, the task is considered successfully solved and sequence of all the corresponding relevant rules form the desired inference path.

5. If some of the required parameters have not been found in the previous step, further search is made. The iterative approach continues searching for rules that can be activated using newly calculated values from previous steps. If there are no such rules, the path does not exist, and the action to be made is similar to stage 2 of the described method. If new

rules that can be activated are present, the algorithm continues. For example, if rule number 2 becomes available, the cell *(2, N +1)* is updated with " ✔" mark (Table 7, Figure 11).

Table 7. Mivar net of the subject domain after the fifth stage

Parameters / **Rules**	1	2	3	4	5	...	*N-2*	*N-1*	*N*	*Service column*
1	X	X	X					Y	Y	✔✔
2			X	Y	Y			X	X	✔
…										
M		X		X	X		Y			
Service row	Z	Z	Z				W	Z	Z	

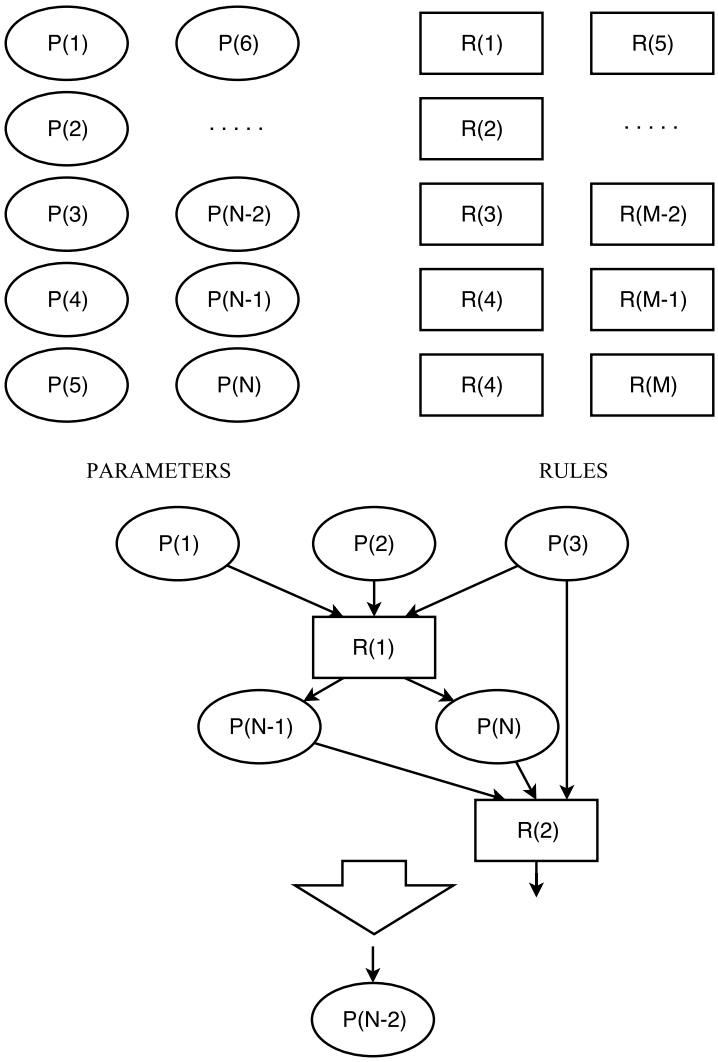


Figure 11. Bipartite graph of inference process: fifth stage

6. Steps 3-4-5 are repeated iteratively until problem is solved. The end result can be either positive - the inference path exists, or negative - no inference path is found due to uncertainty of input data. For clarity, stepping through the example continues where it is necessary to conduct a simulation run of rule number *2* (Table 8, Figure 12).

Table 8. Mivar domain net of the subject domain after simulation run for rule *2*

Parameters / **Rules**	1	2	3	4	5	...	*N-2*	*N-1*	*N*	*Service column*
1	X	X	X					Y	Y	✓✓
2			X	Y	Y			X	X	✓✓
…										
M		X		X	X		Y			
Service row	Z	Z	Z	Z	Z		W	Z	Z	

.

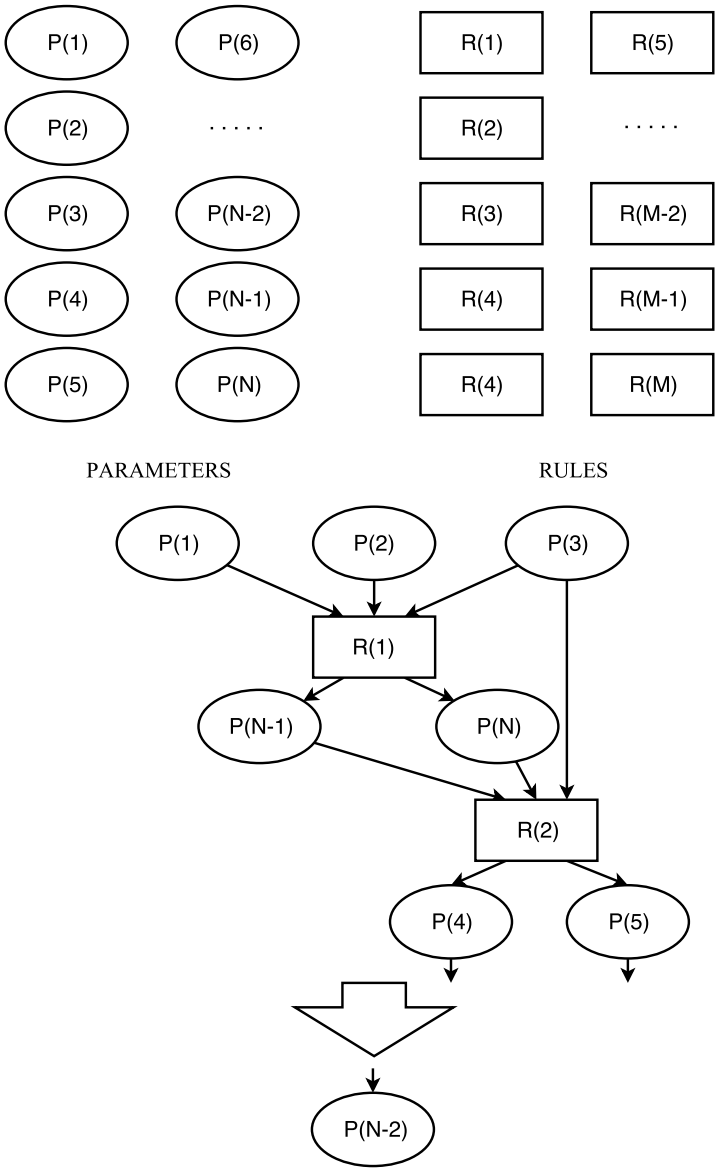


Figure 12. Bipartite graph of inference process: sixth stage

As shown in Table 8 in cells *(M + 1, 4)* and *(M + 1, 5),* parameters *4* and *5 can be derived*. In cell *(2, N + 1)*, it is indicated that the rule has already been launched by putting the sign

"✔✔". After that, service line analysis is performed to check if there are still unknown required parameters remaining. In the example, it is necessary to continue processing the matrix. Further analysis of the matrix shows the ability to run the rule *M* (Table 9, Figure 13).

Table 9. Mivar net of the subject domain in progress for further processing

Parameters / **Rules**	1	2	3	4	5	...	*N-2*	*N-1*	*N*	*Service column*
1	X	X	X					Y	Y	✔✔
2			X	Y	Y			X	X	✔✔
…										
M		X		X	X		Y			✔
Service row	Z	Z	Z	Z	Z		W	Z	Z	

After simulating the rule *M, there* is no pending rules in the service line and the new values in cells of the table are: a cell *(M, N + 1)* contains " ✔✔" and in the cell *(M + 1, N-2)*, *Z* value replaces *W*. Thus, there is a positive result as the inference path, for the given initial values, does exist (Figure 8).

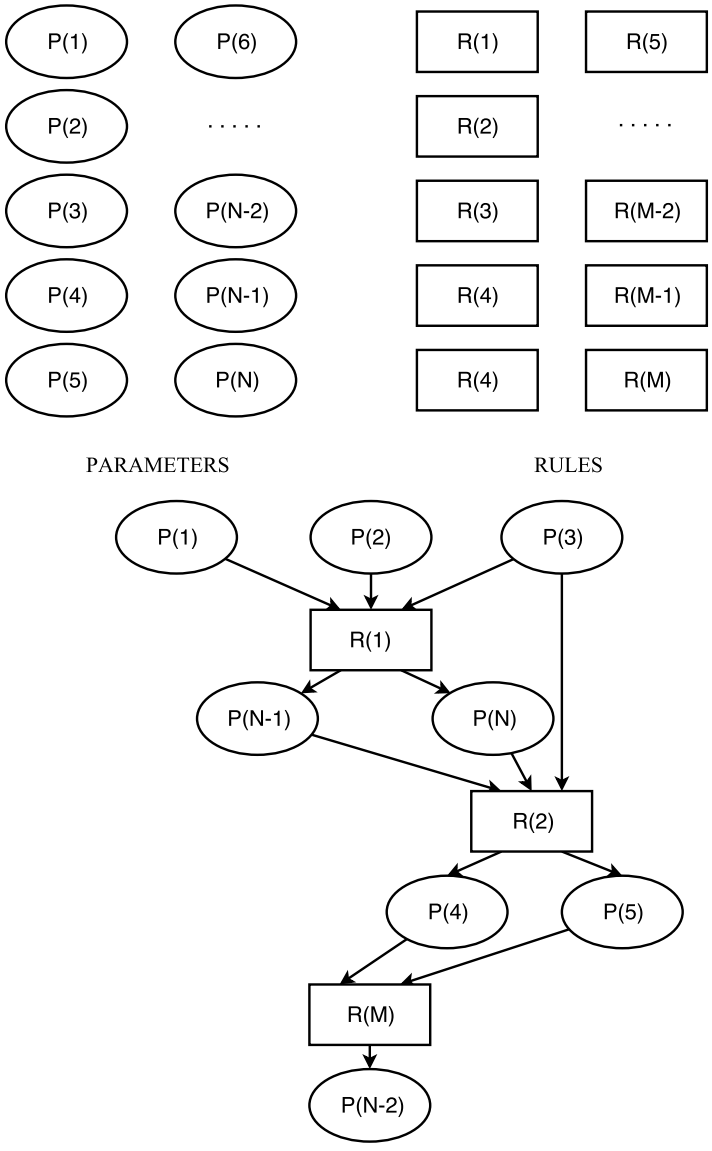


Figure 13. Bipartite graph of inference process: Ready-to-go algorithm

Figure 14 shows a schematic description of the preferred method of implementation for constructing inference path using the proposed mivar approach.

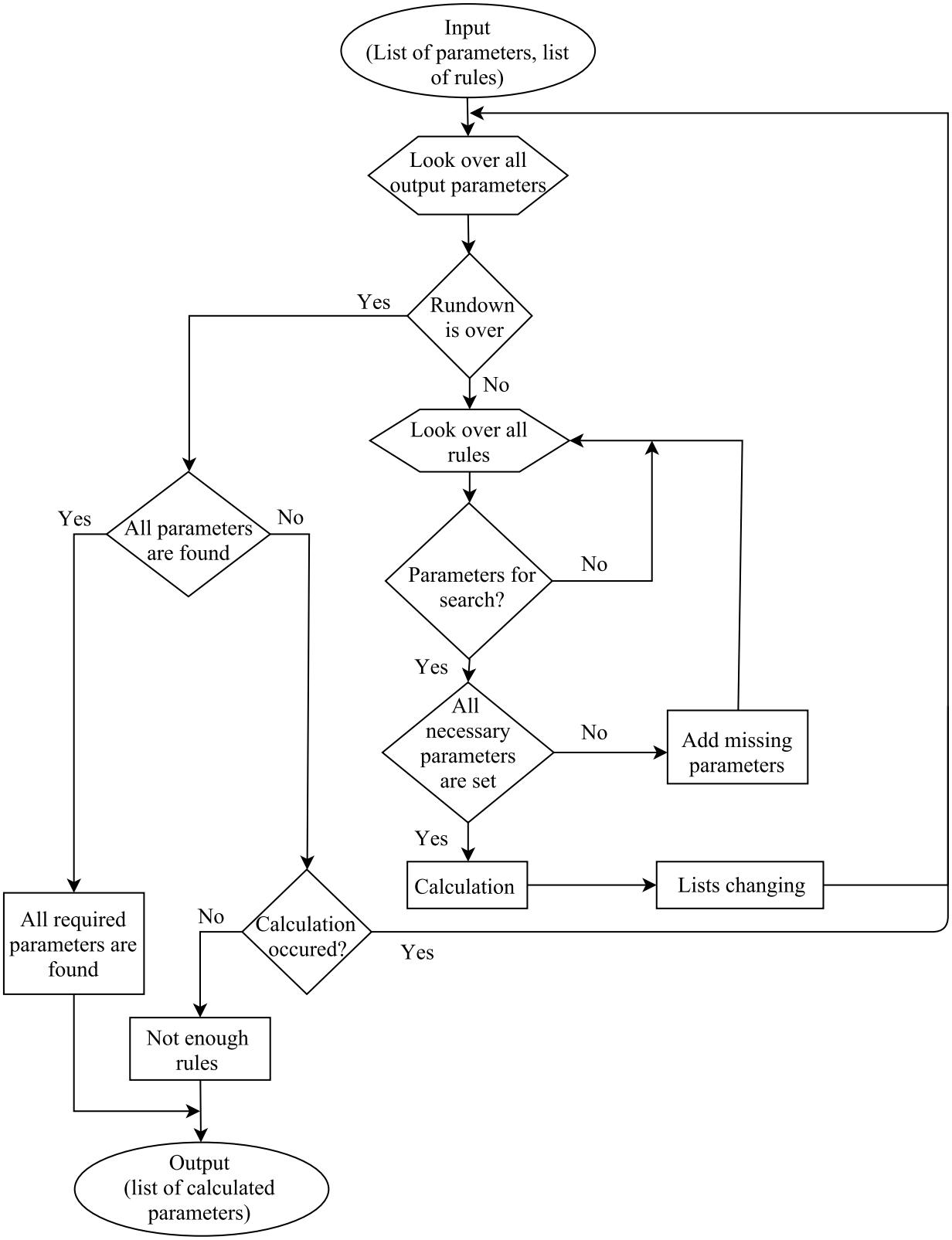


Figure 14. Flowchart of the method of constructing inference path

The above schematics can be illustrated by the following simple example from "Geometry. Triangles" domain. Stages of solving the following task in the given subject area are to be described:

Find length of the sides of ABC triangle, with a perimeter of 16 cm, in which side AB is less than BC by 4 cm and twice less than the side AC.

In the subject area "Geometry. Triangle" a variable may represent various properties such as sides, angles or segments of a triangle. Relationships between these variables (definitions, theorems, axioms, *etc.*) act as rules. Mivar net for solving the problem in the given subject area is given in the form of the matrix M (Table 10).

Table 10. "Geometry. Triangles" example: part of mivar net

Parameters / **Rules**	Side *AB*	Side *BC*	Side *AC*	Perimeter *P*	...	*AB* is (…) less than *BC*	*AB* is (…) times less than *AC*
Perimeter of the triangle (using lengths of the three sides)	**X**	**X**	**X**	**Y**			
The side *AB* using the perimeter and the ratio *AB* and *AC; AB* and *BC*			**Y**	**X**		**X**	**X**
…							
The side *AC* using ratio of the sides and length of the side *AB*	**X**		**Y**				**X**
The side *BC* using ratio of the sides and length of the side *AB*	**X**	**Y**				**X**	

From parsing definition of the discussed problem, perimeter of the triangle and the relationships between pairs of sides of the triangle act as input parameters. The task is to find the sides *BC* and *AC* of *ABC* triangle. This task can be reflected into a working mivar matrix. Small overhead is incurred to the matrix by adding service row and service column. As it has been already mentioned above, the additional matrix row is designed to track changes in the known data. The additional column is designed to track rules usage. Result of the first stage of the work using mivar matrix is presented in Table 11.

Table 11. "Geometry. Triangles" example: results of the first stage

Parameters / **Rules**	Side *AB*	Side *BC*	Side *AC*	Perimeter *P*	...	*AB* is (…) less than *BC*	*AB* is (…) times less than *AC*	*Service column*
Perimeter of the triangle (using lengths of the three sides)	**X**	**X**	**X**	**Y**				
The side *AB* using the perimeter and the ratio *AB* and *AC; AB* and *BC*	**Y**			**X**		**X**	**X**	
…								
The side *AC* using ratio of the sides and length of the side *AB*	**X**		**Y**				**X**	
The side *BC* using ratio of the sides and length of the side *AB*	**X**	**Y**				**X**		
Service row		**W**	**W**	**Z**		**Z**	**Z**	

From the above mentioned matrix, the rule "The side *AB* using the perimeter and the ratios between *AB* and *AC*; *AB* and *BC*" can be launched. That is marked in the respective service rule column cell, and the rule output is marked as a known variable. Results of this stage are shown in Table 12.

Table 12. "Geometry. Triangles" example: results of the second stage

Parameters / **Rules**	Side *AB*	Side *BC*	Side *AC*	Perimeter *P*	...	*AB* is (…) less than *BC*	*AB* is (…) times less than *AC*	*Service column*
Perimeter of the triangle (using lengths of the three sides)	X	X	X	Y				
The side *AB* using the perimeter and the ratio *AB* and *AC; AB* and *BC*	Y			X		X	X	✓✓
…								
The side *AC* using ratio of the sides and length of the side *AB*	X		Y				X	
The side *BC* using ratio of the sides and length of the side *AB*	X	Y				X		
Service row	Z	W	W	Z		Z	Z	

In this way, the parameter *AB* becomes known, and the rule "The side *AC* using the ratio of the sides and the length of the side *AB*" can be run as clear from Table 13.

Table 13. «Geometry. Triangles» example: results of the third stage

Parameters / **Rules**	Side *AB*	Side *BC*	Side *AC*	Perimeter *P*	...	*AB* is (…) less than *BC*	*AB* is (…) times less than *AC*	*Service column*
Perimeter of the triangle (using lengths of the three sides)	**X**	**X**	**X**	**Y**				
The side *AB* using the perimeter and the ratio *AB* and *AC; AB* and *BC*	**Y**			**X**		**X**	**X**	✓✓
…								
The side *AC* using ratio of the sides and length of the side *AB*	**X**		**Y**				**X**	✓✓
The side *BC* using ratio of the sides and length of the side *AB*	**X**	**Y**				**X**		
Service row	**Z**	**W**	**Z(W)**	**Z**		**Z**	**Z**	

Since not all the required parameters have been found, the process continues. However; now *the* parameter *AC* is known. Thus; the rule "The side *BC* using the ratio of the sides and the length of the side *AB*" can be run as shown in Table 14.

Table 14. «Geometry. Triangles» example: results of the fourth pass

Parameters / **Rules**	Side *AB*	Side *BC*	Side *AC*	Perimeter *P*	...	*AB* is (…) less than *BC*	*AB* is (…) times less than *AC*	*Service column*
Perimeter of the triangle (using lengths of the three sides)	**X**	**X**	**X**	**Y**				
The side *AB* using the perimeter and the ratio *AB* and *AC; AB* and *BC*	**Y**			**X**		**X**	**X**	✓✓
…								
The side *AC* using ratio of the sides and length of the side *AB*	**X**		**Y**				**X**	✓✓
The side *BC* using ratio of the sides and length of the side *AB*	**X**	**Y**				**X**		✓✓
Service row	**Z**	**Z(W)**	**Z(W)**	**Z**		**Z**	**Z**	

As a result, an algorithm has been obtained to solve this problem:

«The side AB using perimeter and ratios between AB and AC; AB and BC» -> «The side AC using ratio of the sides and length of the side AB» -> « The side BC using ratio of the sides and length of the side AB».

Next is considering another example of the algorithm derivation for solving a new problem using mivar inference mechanism. The task to be considered is the task of managing the behavior of a multi-agent robotic system. This requires employing two types of robots, namely robots-cooks and robots-waiters in order to organize functionality of a small canteen. It also requires assigning the roles such as preparing incoming order or delivering an order as clear from Figure 15.

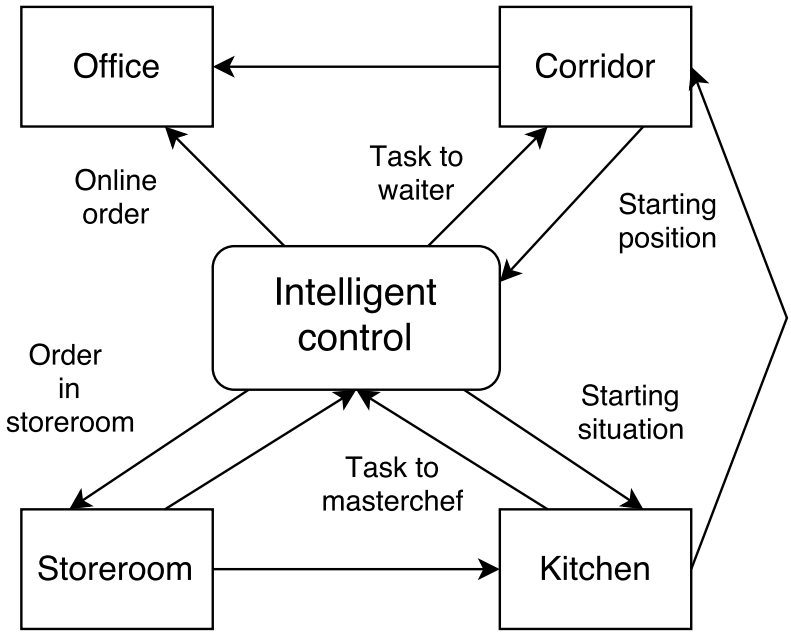


Figure 15. Scheme of a robotic canteen

It has been assumed that management group level is centralized. Moreover; it has been assumed that number of input parameters is limited, movement map (a map with static obstacles) is known, and there is no time limit for order fulfillment. Algorithm of the work is not initially set and it is constructed on basis of the order parameters and the initial conditions. Presence, location and other state parameters for all objects are set at the beginning of calculations. Variations in robot actions depend on the above mentioned initial conditions. For order execution, one unoccupied robot-chef is assigned. If lack of objects is detected in the kitchen, an order is formed for fetching the necessary object from the warehouse. Part of mivar net which arises when considering this subject area is presented in Table 15.

Table 15. Robotic café example: part of mivar net

Parameters / **Rules**	Empty teapot	Cold teapot	Hot teapot	Water	Green tea in the cupboard	Green tea on the table	...	Cup	No green tea	Green tea order
Pour water into the teapot	X	Y		X						
Boil the teapot		X	Y							
…										
Brew tea			X			X		X		Y
Get the tea leaves out of the closet					X	Y			X	

If the task is to brew green tea. It is well-known that it needs water, cup, green tea in the closet, and an empty kettle. In the first stage, the known data in the extended mivar matrix is noticed as clear from Table 16.

Table 16. Robotic café example: first stage results

Parameters / **Rules**	Empty teapot	Cold teapot	Hot teapot	Water	Green tea in the cupboard	Green tea on the table	...	Cup	No green tea	Green tea order	*Service column*
Pour water into the teapot	X	Y		X							
Boil the teapot		X	Y								
…											
Brew tea			X			X		X		Y	
Get the tea leaves out of the closet					X	Y			X		
Service row	Z			Z	Z			Z	Z	W	

It can be seen that the rule «Pour water into the teapot» can be activated during the first step. Outcome of this step is the appearance of a cold teapot. Next action which can be performed is «Boil the teapot». Appropriate actions are presented in Table 17.

Table 17 Robotic café example: results of the second and the third stages

Parameters / **Rules**	Empty teapot	Cold teapot	Hot teapot	Water	Green tea in the cupboard	Green tea on the table	...	Cup	No green tea	Green tea order	*Service column*
Pour water into the teapot	X	Y		X							✓✓
Boil the teapot		X	Y								✓✓
…											
Brew tea			X			X		X		Y	
Get the tea leaves out of the closet					X	Y			X		
Service row	Z	Z	Z	Z	Z			Z	Z	W	

Acting in a similar way, the following sequence of actions can be obtained: *«Pour-water into the teapot» -> «Boil the A teapot» -> «Get the tea leaves out of the closet» -> «Brew tea».*

It should be noticed that during deriving algorithm of actions, some solutions may contain unwanted superfluous steps. In most cases, presence of such extra steps is not critical for calculation time. However; in order to further accelerate the calculations, the inference mechanism implemented in Wi!Mi is supplemented by a block which eliminates all unnecessary calculation steps. For this purpose, each detected parameter contains information about the launched rule to define this parameter. On the other hand, each rule is supplemented with information about “parent” parameter.

10.7 Computational complexity of mivar method

*O. O. Varlamov, Wi!Mi Expert System Shell as the Novel Tool for Building Knowledge-Based Systems with Linear Computational Complexity. International Review of Automatic Control. **11**(6), 314-325 (2018).*

Testing the ES created by Wi!Mi has been conducted for

a) Real subject domains.

b) Randomly generated mivar nets.

Random generation of mivar nets is necessary to test computing speed for larger mivar nets where number of rules exceeds 5000. The tests have been carried out on a PC with the following specifications: ASUS Maximus V Extreme motherboard with Intel® Core ™ i7-3770K processor, 8GB of Corsair DDR3 PC12800 RAM, NVIDIA GeForce GTX 670 2048M graphics card, 120GB Intel 520 SSD, and Windows 7 x64 as operating system. The area "Geometry. Triangle" has been chosen for real domain tests. It could be described as a 33 × 161 mivar net. Searches for inference path to solve various problems in such a mivar net have taken less than 5 ms.

Further testing of the prototype has been carried out on a variety of subject areas similar to real life domains. Number of rules described in the subject area " Geometry. Triangle" was approximately three times greater than number of parameters. To generate a random mivar net, number of the rules has been set and number of the parameters has been calculated as: <number of rules> / 3.

Detailed testing results are given in Tables 18 and 19.

Footprint and performance results are presented in Table 1 – regular demanding requirements ((it is needed to find 0.03% of all parameters existing in the considered subject domain)). Table 2 contains heavy demanding requirements. These requirements are needed to find 80% of all the parameters which exist in the considered subject domain.

All the tests have been carried out on a testing model. In those generated random models, number of rules have been set and number of objects (parameters) have been calculated as: <number of rules> / 3. A prototype of this system has been taken as ES in the subject domain Geometry. In this subject area, a number of rules is approximately three times greater than the number of parameters.

Table 18. Tests of mivar net productivity while increasing number of rules. Number of rules varies between 400,000 and 1,000,000 with increments of 100,000. Example with 3,000,000 rules is also shown. 0.03% of total number of the parameters were defined at the end of each run.

Number of rules in ES	Number of parameters found	Time to build inference tree in ms	Memory footprint MB
400000	133	1	142
500000	166	1,45	181
600000	200	3,02	218
700000	233	1,71	255
800000	266	2,6	293
900000	300	4,6	322
1000000	333	3,7	386
3000000	1000	14,04	1056

Table 19. Tests of mivar net productivity while increasing number of rules. Number of rules varies between 400,000 and 1,000,000 with increments of 100,000. Example with 3,000,000 rules is also shown. 80% of the parameters have been defined during each run.

Number of rules in ES	Number of parameters found	Time for breadth-first search in ms	Time to build inference tree in ms	Memory footprint MB
400000	106666	2762,01	934,19	199
500000	133334	4112,68	1217,19	181
600000	160000	5923,57	635,51	218
700000	186666	7664,87	1768,87	255
800000	213334	10038,80	1148,22	293
900000	239997	12838,90	891,93	322
1000000	266665	15547,70	1162,70	386
3000000	800000	135571	3608.99	1539

Computational complexity of the proposed algorithm can be theoretically estimated. Assuming that one rule produces single new parameter, the worst case scenario can be roughly estimated for Wi!Mi inference search algorithm. In each iteration, at least one new parameter can be defined. Thus, all parameters are defined in a well-built MIVAR net after n steps. On the other hand, m rules at most must be scanned during each iteration. Therefore; computational complexity of the search for an algorithm in the worst case is proportional to mn or $O(mn)$ (where m is number of rules and n is number of parameters). An estimate for breadth-first search which is used for Mivar nets traversal gives similar results, *i.e.* $O(|V| + |E|)$ where $|V|$ is number of vertices, while $|E|$ is number of edges. In the worst case scenario, every vertex and every edge must be visited. $|V| = m + n$ for bipartite graph and if graph is sparse then $|E| \sim (m + n)$. Therefore; the overall complexity is virtually linear with time complexity $O(m + n)$.

Additional remarks

Wi!Mi is a cross-platform software but it requires individual building or compilation for each platform that it supports.

To build complex rules (connections between objects) in ES developer, Java Script can be used.

Wi!Mi is able to handle 3 000 000 rules and construct an algorithm for solving a problem in less than four seconds. This is nearly 100 times faster than what was achieved with the earlier prototypes of the system (Varlamov, Linear matrix method for determining a route on the adaptive network of rules, 2002a).

MIVAR method of logical inference was published in the research paper:

O. O. Varlamov, Wi!Mi Expert System Shell as the Novel Tool for Building Knowledge-Based Systems with Linear Computational Complexity. International Review of Automatic Control. 11(6), 314-325 (2018).