Adaptive control of modular robots

Demin A. V. Vityaev E.E.

Institute of Informatics Systems Sib.Br.RAS, pr. Lavrentieva 6, Novosibirsk, Russia
Sobolev Institute of Mathematics pr. Koptyuga 4, Novosibirsk, Russia

Astarct. *This paper proposes a learning control system of modular systems with a large number of degrees of freedom based on joint learning of modules, starting with finding the common control rules for all modules and finishing with their subsequent specification in accordance with the ideas of the semantic probabilistic inference. With an interactive 3D simulator, a number of successful experiments were carried out to train three robot models: snake-like robot, multiped robot and trunk-like robot. Pilot studies have shown that the approach proposed is quite effective and can be used to control the complex modular systems with many degrees of freedom.*

Key words: control system, patterns detection, knowledge elicitation

1. Introduction

The task of developing control systems for modular robots faces serious difficulties resulting from the hyper-redundancy of such systems. The existing approaches based on the reinforcement learning techniques are not effective for systems with a big number of degrees of freedom, and evolutionary methods have severe limitations associated with the necessity of the robot population availability, which does not enable learning and adaptation in real time.

The purpose of the paper is to develop the control systems for hyper-redundant modular systems which enable learning and adaptation of such systems in real time. To meet this challenge, it is proposed to use logical-and-probabilistic methods of knowledge elicitation adapted for control purposes.

Results. This work proposes a new approach to creating the learning control systems for modular robots, based on the use of the modules functional similarity and the logical-and-probabilistic algorithm of the guided search of rules. The approach is based on the joint learning of the control modules, starting with finding the common control rules for all modules and finishing with their subsequent specification in accordance with the ideas of the probabilistic inference. The main advantage of the proposed approach is the high learning rate and the teach-and-learn capability in real-time mode based only on the experience of the system's interaction with the environment. The paper gives the examples of creation and learning of control systems for three virtual robot models: snake-like robot, multiped robot and elephant's trunk robot. The experiments carried out have confirmed the learning high rate and quality of control. Practically speaking, the results of the experiments show that the proposed approach to adaptive control can be used in the tasks of control systems development by intelligent agents - software or robotic systems, including hyper-redundant, which require learning capability and adaptation to changing circumstances.

2. Modular Robots

Currently, the new area in robotics under the name "modular robots" [1,2] is developing actively. The basic idea of this approach is the robotic engineering from a variety of the simple, single-type modules that themselves have limited locomotion, but by connecting with each other can form complex mechanical systems with a big number of degrees of freedom.

Such robots have a number of interesting possibilities exceeding the abilities of traditional robots. Firstly, it is the ability to create different designs from the same modules, that enables solving different tasks using the same set of modules. It is much more cheaper and more convenient than constructing a lot of specialized robots for each specific task. Moreover, it is possible to create robots-transformers that independently change their design, depending on the environmental objectives and conditions. Second, the modular structure and availability of a large number of degrees of freedom (hyper-redundancy) allows one to create fault-tolerant robot models. Such robot's individual modules' failure

is not critical to the operation of the entire system, and causes minimal performance degradation Thirdly, the production and use of such robots is economically advantageous since the modules of the same type are simpler and cheaper to manufacture and repair.

However, the broad capabilities of modular robots associated with hyper-redundancy also have the downside - significant complexity of control. In particular, the relevant task is creating a locomotion control system for a predefined robot configuration. Whereas for usual robots the traditional approach to creating control systems is the manual programming by a man-developer, for modular robots this approach is inefficient. Because of the large number of degrees of freedom it is extremely difficult for a developer to foresee and to program all the possible forms of movement and the situations where they need to be applied, and particularly to take into account the ability of adaptation in the event of individual modules' failure or a sudden environment change. Therefore, it becomes relevant to develop the ways of control system automatic generation based on different learning models.

However, the use of popular methods such as reinforcement learning, for generation of control systems of hyper-redundant robots is difficult due to the large number of degrees of freedom of such robots. Thus currently many developers generally prefer using the evolutionary method, in particular, genetic algorithms and genetic programming, as well as their combinations with standard learning methods [3-9].

But the application of evolutionary methods also has drawbacks, the main ones of which are as follows [10]. First, this is a significant time required to perform calculations because at each evolutionary step, each solution from the population requires the control quality assessment for the purpose of which each solution is required to be executed. Secondly, using this method makes adapting to the real work conditions virtually impossible, because the method requires the availability of a population of robots

In this paper, we propose a learning control system using the logic-probabilistic approach to knowledge elicitation for the monitor rules generation from the system's environment interaction experience [11-15,18]. The specificity of the suggested approach is that the system is primarily trying to locate the monitor rules, common to all modules, and only then - the rules that are specific to each individual module. The effectiveness of the approach is suggested to evaluate by the example of teaching the typical representatives of simple hyper-redundant modular robots: snake-like robot, multiped robot and trunk-like robot.

3. Simulator

For conducting the experiments with the proposed control model an interactive 3D-simulator with graphical user interface was developed. The main purpose of the program is to conduct experiments on robot control in the environment pushed closer to the real world. The program has virtual environment visualization capabilities as well as the capability of recording experiments in a video file. As a physics engine, the simulator uses the Open Dynamic Library (ODE) [16], which allows you to simulate the dynamics of solid bodies with different types of joints. The advantage of this library is the speed, high stability of integration, as well as built-in collision detection. Using this simulator, three robot models were built: snake-like, multiped and trunk-like robot.

Figure 1. Snake-like robot model

Snake-like robot model is presented in the simulator as a set of six identical rectangular blocks ("vertebrae") combined by universal joints (Figure 1). All joints are identical and have two angular engines ("muscles") that ensure the rotation of the joints in the vertical and horizontal planes. The proposed design, despite its simplicity, provides sufficient flexibility of the model and provides body's position typical for the biological snakes.

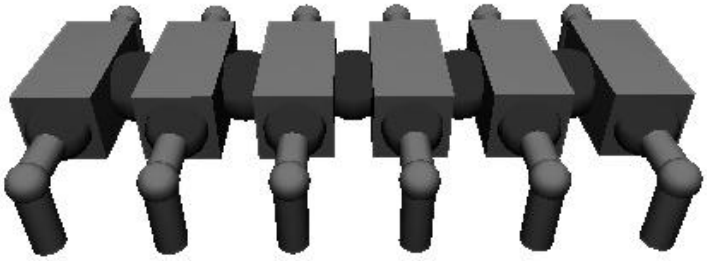

Figure 2. Multiped robot model

The second model, a multiped robot, is presented as a structure of six identical modules connected to each other with rigid joints (Figure 2). Each module has a pair of L-shaped legs on the right and left side. So, the robot has totally twelve legs-limbs. Each leg is connected to the module by a universal joint with two angular motors allowing a joint to rotate the leg in the horizontal and vertical planes. In general, the robot's design resembles its biological type of millipede, and allows you to implement specific ways of movement.

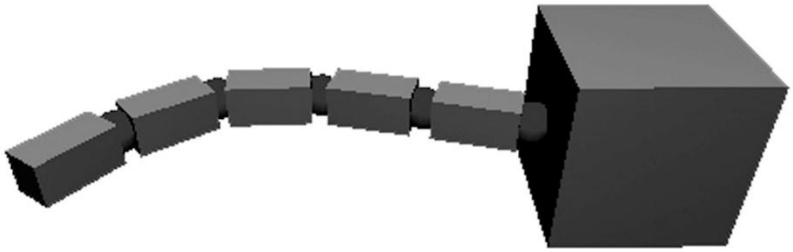

Figure 3. Trunk-like robot model.

Model of trunk-like robot is represented in the form of a multi-segmental "trunk", connected by a versatile joint with a massive fixed platform in the form of cube. The trunk itself is represented as a sequence of five identical rectangular blocks associated with versatile joints provided with angular motor. The dimensions of the blocks and the position of the joints have been chosen in such a way as to ensure the system's sufficient flexibility and attainability to carry out the experiments.

4. Control System

To create the modular robots control system, it is proposed to use neural networks consisting of trainable logic neurons, each of which controls a separate robot module.

Logical neurons operate in a discrete time $t = 0,1,2,\ldots$. Each neuron contains a set of inputs $input_1,...,input_k$ assuming valid values and one output $output$ assuming the values from a predefined set $\{y_1,...,y_m\}$. At any timepoint t, the neuron inputs are supplied with information by assigning real values of the inputs $input_1 = x_1,...,input_k = x_k$, $x_1,...,x_k \in \mathbb{R}$. The result of the neuron's work is the output signal $output = y$, $y \in \{y_1,...,y_m\}$ assuming one of the possible values $\{y_1,...,y_m\}$.

After all the neurons of the network have completed the work, the reward comes from the external environment The award function is set depending on the ultimate goal and is used to evaluate the control quality. Control system task is detecting such patterns of neuron functioning that would ensure getting the maximal reward.

Variety of the patterns that define the work of neurons are suggested to search in the form of logical patterns with estimates that are as follows:

$$\forall i\,(P(i), X_1(i),..., X_m(i), Y(i) \to r)\,, \qquad (1)$$

where $i = 1,..,n$ - the variable on the neurons.

$X_j(i) \in \mathbb{X}$ - predicates from the specified set of input predicates $\mathbb{X}$ that describe the inputs j of the neuron N_i $(i = 1,..,n)$. For example, in the simplest case, these predicates can be preset as $X_j(i) = (input_k(i) = x_r)$, where x_r - some constants from the range of input signal values that can be preset, for example, by quantization the range of possible values of the corresponding inputs of neurons.

$Y_j(i) \in \mathbb{Y}$ - predicates from a given set of output predicates $\mathbb{Y}$ describing the output of neurons N_i $(i = 1,..,n)$ and looking as $Y_j(i) = (output(i) = y_r)$ where y_r - some constants from the range of output signals.

$P(i) \in \mathbb{P}$ - predicates from a set of predicates $\mathbb{P}$ look as $(i = j)$, where $j = 1,..,n$ the intent of which is to narrow the scope of type rules application (1) to specific neurons.

r - reward whose maximization is the task of a neuron.

These patterns predict that if neuron gets input signals $N_i, i = 1,..,n$ that satisfy the input predicates $X_1(i),...,X_m(i)$ of the rule premises, and the neuron sends output signal specified in the output predicate $Y(i)$, the reward mathematical expectation will be equal to a certain value r.

Additionally, we will note that if a neuron N_j has an input specific only to that neuron, it is assumed that the predicate $X(i)$ describing this input will take the value "0" for all $i \neq j$, i.e. for all other neurons. Similarly, if the output of any neuron N_j can take any value y specific only to that neuron, the corresponding output predicate $(output(i) = y)$ will also take the value of "0" for all $i \neq j$.

We will now explain the need for introducing a set of predicates $\mathbb{P}$. Should a rule (1)does not contain predicates from $\mathbb{P}$, it will look like $\forall i(X_1(i),...,X_m(i),Y(i) \to r)$ and will describe patterns common to all neurons $N_i, i = 1,..,n$. Adding a predicate to premises of the rule from $\mathbb{P}$ automatically narrows the scope of the rule application to a specific neuron. Thus, the rules containing predicates from $\mathbb{P}$, describe patterns specific to particular neurons. It should also be noted that the narrowing of the rule scope applicability (1)can takes place not only because of predicates from $\mathbb{P}$, but also because of input or output predicates from $\mathbb{X}$ and $\mathbb{Y}$ describing specific inputs or outputs of particular neurons.

In order to find the patterns of the type (1), it is proposed to use an algorithm based on the semantic probabilistic inference described in the papers [13.17]. This algorithm helps to analyze the data set that stores the statistics of the neural network operation (input-output of neuron and the reward received) and all the statistically significant patterns of the type (1) are extracted.

Let's consider the algorithm of patterns search in more detail.

Further, for simplicity, we'll write down the rules (1) in simplified form:

$$P, X_1,..., X_m, Y \to r. \qquad (2)$$

We will introduce some formal definitions.

Let's rewrite the rule (2)in the form $A, Y \to r$ where A indicates predicate set from the set of $\mathbb{P}$ or $\mathbb{X}$ that are included in the rule premise, i.e $A = \{P, X_1,..., X_m\}$.

Definition 1. *Subrule* of the rule $R_1 = A_1, Y \to r$, $A_1 \neq \varnothing$ we'll refer to any rule $R_2 = A_2, Y \to r$ for which the condition has been met $A_2 \subset A_1$.

Definition 2. *The pattern* we will call the rule of type (2)meeting the following conditions:

1) The mathematical reward expectancy r for the rule is defined.

2) The mathematical reward expectancy r of the rule is strictly more than the mathematical reward expectancy for each of its subrules.

Definition 3. The rule $R_2 = A_2, Y \to r$ will be called *rule* refinement $R_1 = A_1, Y \to r$ if one of the conditions is executed for it

1) $A_2 = A_1 \cup X$, where $X \in \mathbb{X}$ and $X \notin A_1$,

2) $A_2 = A_1 \cup P$, where $P \in \mathbb{P}$ and A_1 contains only predicates from $\mathbb{X}$.

Thus, the rule R_2 is a refinement of the rule R_1 if it is obtained either by adding a new predicate to the premise from $\mathbb{X}$ or by adding any predicate from $\mathbb{P}$ if in R_1 there are no predicates of this type. The essence of operation refinement is to specify the rule scope applicability, either by adding new input features or by narrowing the rule applicability to a particular neuron.

The essence of the algorithm for detecting patterns is a sequential refining the rules, starting with the rules of the unit length, by adding new the predicates to the rules premise succeeded by checking the refined rules for belonging to probabilistic patterns. Essentially, the directional search of rules that can significantly reduce the search space is implemented. Search reduction is achieved through the use of heuristics, which is that, starting from the moment when the length of the rules premise reaches a certain preset value, called the depth of basic search, only the rules that are patterns are refined successively.

Let's now proceed to the description of the algorithm that implements the search for many patterns defining the behavior of neurons. Let us denote $Spec(RUL)$ - the set of all the rules possible refinements from RUL where RUL is an arbitrary set of type rules(2). The input parameter of the algorithm is the depth of the basic search d, where $d \geq 1$ is the natural number.

1. At the first step, we generate a set of RUL_1 of all the rules of unit length represented as $Y \to r$, $Y \in \{Y_1,...,Y_k\}$. All the rules of RUL_1 are checked for the conditions of belonging to the patterns. The rules that have been verified will be considered to be patterns. Let's denote through REG_1 the set of all the patterns found at the first step.

2. At step $k \leq d$ the set $RUL_k = Spec(RUL_{k-1})$ is generated of all the rule refinements having been generated during the previous step. All the rules from RUL_k are checked for the conditions of belonging to the patterns. Let's denote through REG_k the set of patterns received.

3. At step $l > d$ the set $RUL_l = Spec(REG_{l-1})$ is generated of all the patterns refinements having been found during the previous step. All the rules from RUL_l are checked for the conditions of belonging to the patterns. Let's denote REG_l - the set of all the patterns found at this step.

4. The algorithm stops at the step $m > d$ when no new patterns are found $REG_m = \varnothing$.

5. The resulting set of patterns is the union of all the sets of patterns found REG_i.

The algorithm $k \leq d$ steps correspond to the basic search and steps $k > d$ - to the extra search.

Evaluation of reward expectation for rules is carried out according to data set that stores the statistics of the system operation (input / output of neurons and the reward received), as follows: $r = \sum_{i \in I} r_i \Big/ |I|$, where I is the set of events when, as a rule, can be applied r_i - the neuron's reward for the i-th event $i \in I$.

The advantage of using semantic probabilistic inference and type rules (1)is in organizing the rules search in such a way that the rules common to all neurons will be discovered first, and then more complex rules including specific ones for particular neurons. As a result, the suggested approach can significantly reduce the solution search time of the tasks of modular robot control if at least part of the modules have similar functions that can be described by general rules.

The behavior of the neural network as a whole is as follows: In each operating cycle of network, the incoming signals are received on the neuron inputs. After that, the decision-making procedure is started successively for each neuron, in course of which from the set of rules describing the neurons behavior, those ones shall be selected that can be applied to the current neuron on the current input signals. A single rule is then selected among the selected rules that forecasts the maximum value of the mathematical expectation of the reward r. Further, the neuron output receives the output signal $output = y$ specified in the rule. At the initial stage of the network functioning, when the set of rules describing the neurons behavior is still empty, or when there are no rules applicable to the current set of incoming signals, the neuron's output is determined randomly. After all the neurons generate their

output signals, the external environment provides a reward and learning in the process of which which the new rules are searched and current rules of operation are adjusted in accordance with the suggested patterns of algorithms search.

5. Snake-Like Robot Locomotion Control System

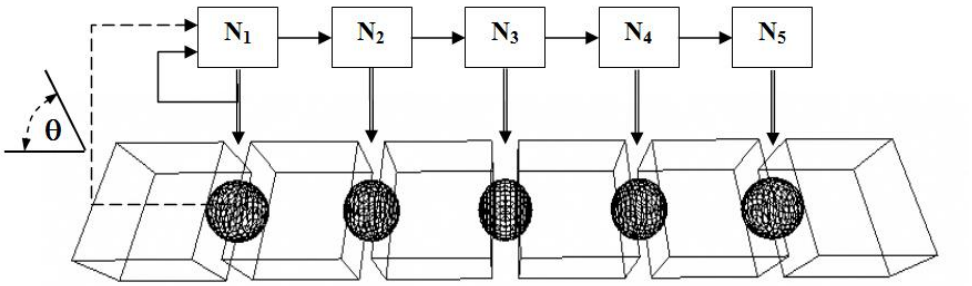

Figure 4. Diagram of the neural control circuit of the robot locomotion

The purpose of this experiment was the teaching the simplest model of the snake-like robot locomotion forward (Figure 1). In previous studies [18.19] we proposed model of the neural control circuit of the robot locomotion of nematode C. Elegans, which proved to be highly effective in experiments on undulating way of locomotion learning process. The diagram of this circuit assumed that the head of nematode acted as a source of oscillations, based only on feedback from stretch receptor. Then the signal is distributed over the nematode's body with a certain time delay, thereby enabling the distinctive undulation. Since the design of the snake-like robot has many commonalities with the nematode model, it was decided in this paper to use a similar neural circuit scheme to control the robot locomotion.

Finally a neural circuit consisting of five neurons (Fig. 4) was selected. Each neuron N_i, $i = 1,...,5$ controls one joint of the robot body, sending activation signals to the angular motors located in that joint. The head neuron N_1 receives input information about the bending angles between the headt and the subsequent segment. In addition to this, the neuron receives a signal from its own output with a time delay Δt via feedbacks. The remaining neurons $N_i, i = 2,...,5$ receive only the signal from the previous neuron's N_{i-1} output with a time delay of Δt.

The set of input and output predicates for neurons is specified by quantizing the range of possible values of the neuron corresponding inputs and outputs. The reward for the entire neural circuit of locomotion control is determined depends on the speed that the robot will develop on the time span of Δt: the higher speed - the greater reward.

A number of successful experiments were conducted using the 3D-simulator on learning of the proposed model for ways of locomotion. As the results of the experiments have shown, the control system has been able to consistently learn an effective way of locomotion forward based on the undulating body movement in the horizontal plane. This way of movement is the most common among the biological snakes and is also typical for some other animals, for example, nematodes. Figure 5 shows the best sequence of motions found by the system during learning when moving forward.

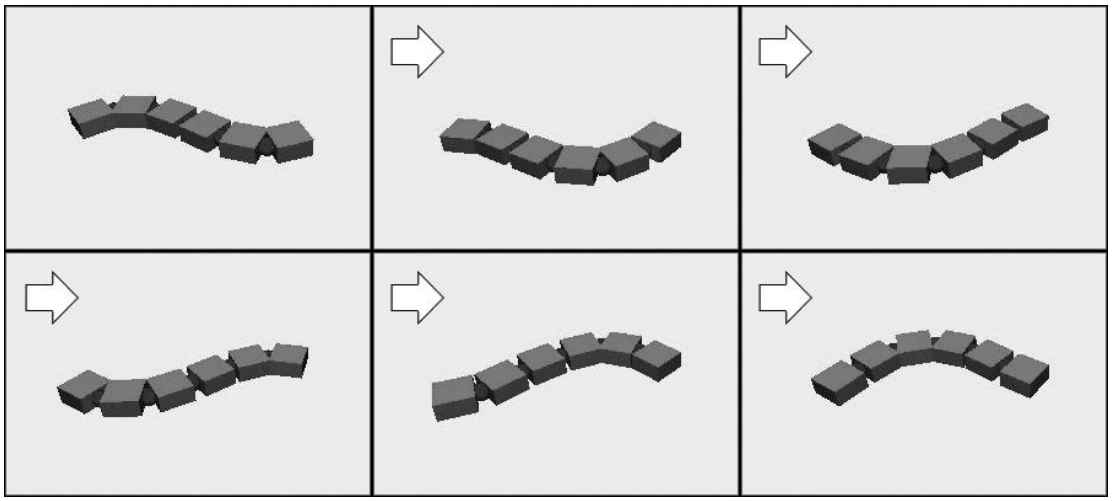

Figure 5. Sequence of the snake-like robot movements when moving forward

6. Multiped Robot Locomotion Control System

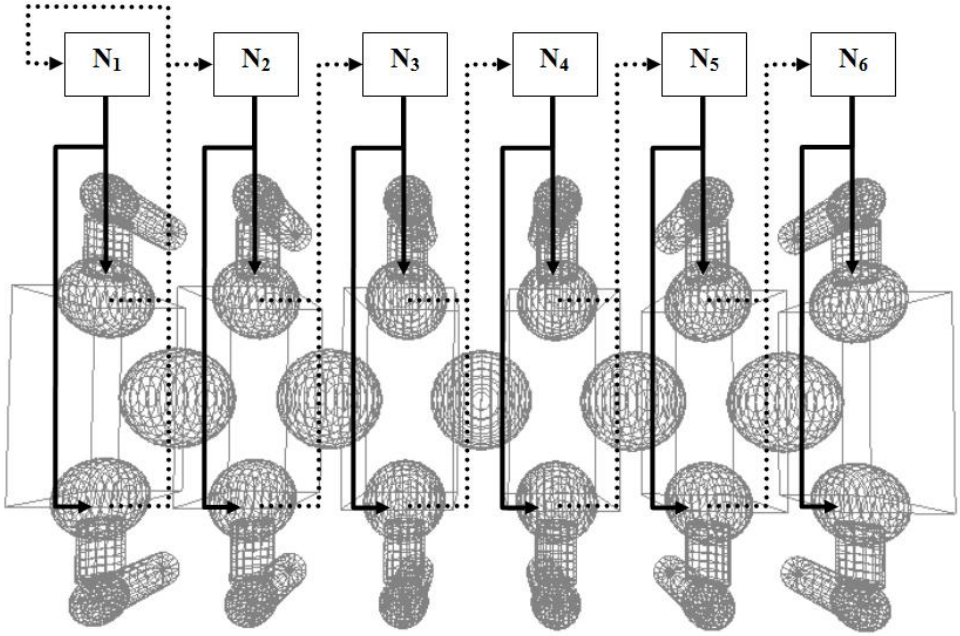

Figure 6. Diagram of the neural control circuit of the multiped robot locomotion

The purpose of this experiment was to teach the multoped robot model (Figure 6) the way of movement forward. For multiped robot control system, the neural circuit consisting of six neurons was selected - one neuron for the robot each module (Figure 6) Each neuron N_i, $i = 1,...,6$ controls the movement of its module's left and right leg by sending out signals to the appropriate angular motors rotating limbs in the joint. To make the task a little easier, the right and left legs of the robot were synchronized so that the movement of one leg always occurs in counterphase to the other, for example, the forward movement of the left foot is always accompanied by the backward movement of the right foot. So the neuron is essentially enough to control the movement of only one leg, because the second leg will repeat the same movements only in counterphase.

Neuron of the first module N_1 gets the incoming information about the first module's leg position. The remaining neurons N_i, $i = 2,...,6$ receive the incoming information about the legs position of the previous module. Information about the position of the legs is set by a pair of angles of the limb

bending in joint in the vertical and horizontal planes. The set of input and output predicates for neurons is specified by quantizing the range of possible values of the neuron corresponding inputs and outputs. The reward for the entire locomotion control system is determined depends on the speed that the robot will develop on the time span Δt of : the higher speed - the greater reward.

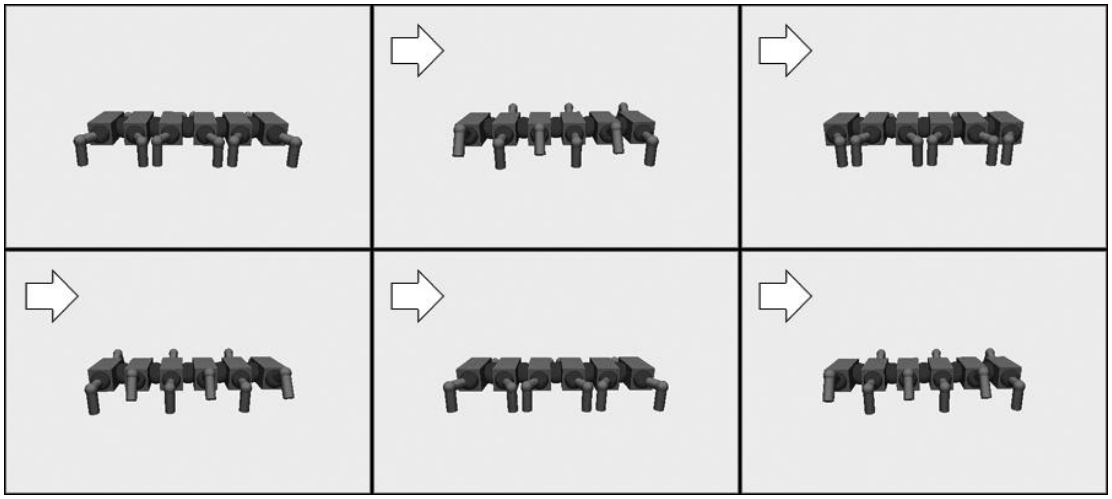

Figure 7. Sequence of the multiped robot movements when moving forward

A number of successful experiments were conducted using the 3D-simulator on learning locomotion of the multiped robot model. The results of the experiments showed that the control system successfully discovers cooperative movements of limbs ensuring effective movement forward. Figure 7 shows an example of the best sequence of motions found during the learning.

7. Trunk-like Robot Locomotion Control System

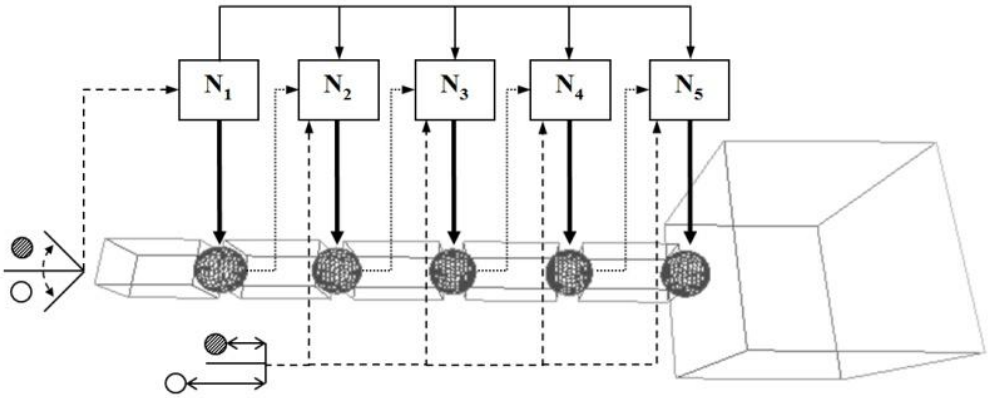

Figure 8. Diagram of trunk-like robot neuron circuit

In this experiment, the robot task (Figure 8) was to grasp a target that appears in a random position within the trunk's range. A certain sphere with a radius equal to the length of the trunk's one segment acts as a target. The target is considered to be grasped if the end of the trunk (the last segment) is found to be inside the sphere. After grasping, the sphere-target disappears and appears in a new random position. Thus, the experiment may continue uninterrupted for unlimited period of time. The purpose of the control system is the detection of such control rules of trunk movement that would ensure the target grasping in any position that is accessible for the manipulator.

To accomplish this task, we have selected the control scheme of five neurons N_i, $i=1,...,5$, each of which controls a single segment of the trunk (Figure 8). Neurons serve as triggers for angular motors, thereby causing the trunk bend in the appropriate joints. Since in this task the target can only be at one level, all motions are limited to a horizontal plane simplifying the task.

The first neuron N_1 receives incoming binary information on which side of the trunk the target is: on the right or left. The rest neurons N_i, $i=2,...,5$ receive the following incoming signals: (1) information about bend angle between the controlled and previous segment (2) signal from the previous neuron's output N_{i-1} at the previous timepoint; (3) binary information about the position of the target with respect to the end of the trunk and its attachment point - the result of comparing the distances from attachment to the target and from attachment to the manipulator end. Thus, in this experiment, the manipulator is actually "blind", that is, it does not see the exact position of the target, but only "feels" it: right-left and closer-farther.

The reward for control system is calculated based on the fact of target grasp by the manipulator, as follows: Suppose the target appeared in a new position at the moment of time t_0, and the manipulator grasped it at the moment of time t_1. Then all actions from the moment of time t_0 to the moment of time t_1 will receive the reward in the amount of $r=1/(1+(t_1-t))$ where t the moment of time for which the reward is calculated. Thus, the most recent actions that has led to the objectives achievement shall receive the biggest reward, and the deeper in the past from the moment of grasp the action took place, the less reward it receives.

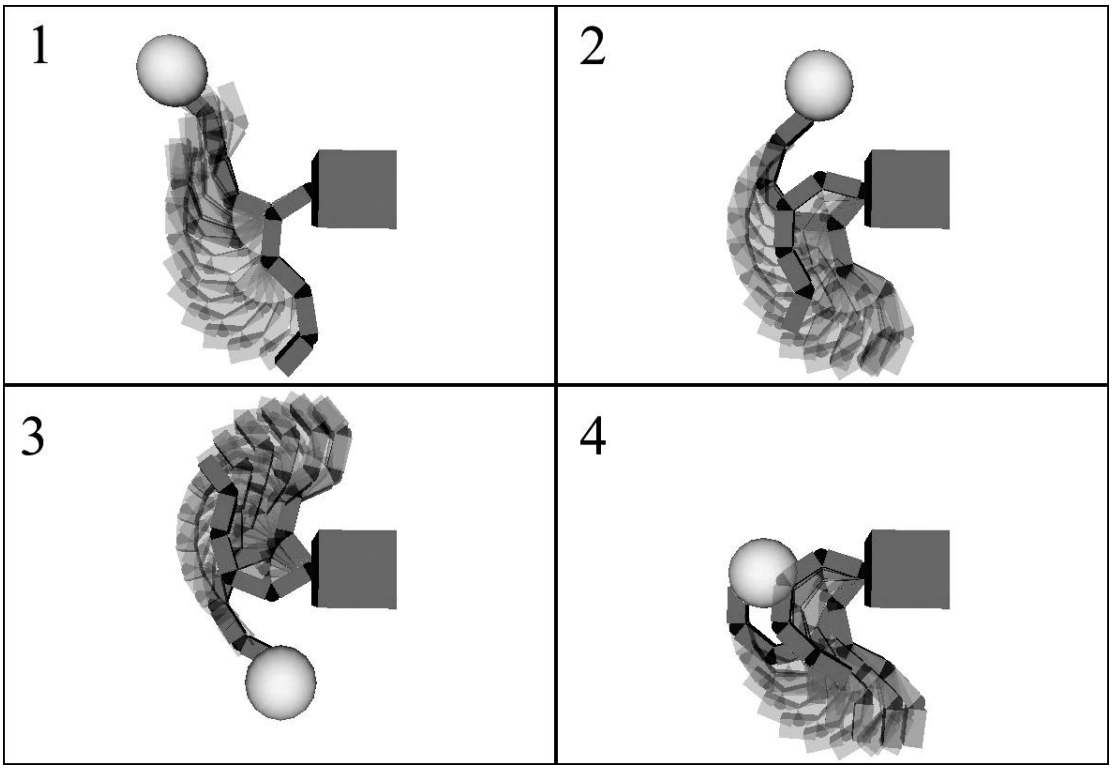

Figure 9. Example of the manipulator path when grasping a target

With a 3D simulator, the experiments were carried out on learning the manipulator to grasp the targets at any position within the manipulator's grasp. The results of the experiments showed, the control system is able to detect effective rules ensuring 100% grasp of targets in the specified zone. Figure 9 shows examples of the paths for already taught trunk when grasping the target. The examples from second to fourth show that the control system has learned to make special preparatory movements to be ready for grasp. In the aforesaid examples, the trunk start bending is directed toward the side opposite of the target, and in order to achieve the target, the manipulator makes the undulating motion in the other direction, which changes the body's inflexion to the opposite one and then successfully grasp the

target. These examples demonstrate that the control rules detected by the system generate rather complex behavior aimed to the goal achievement.

8. Conclusion

This paper proposes an adaptive control of modular systems with a large number of degrees of freedom based on joint learning of control modules, starting with finding the common control rules for all modules and finishing with their subsequent specification in accordance with the ideas of the semantic probabilistic inference. The main advantages of the proposed approach are: teach-and-learn capability in real time and the high learning rate which is achieved through the effective use of the module's functional commonality properties and the algorithm of the rule-oriented search . In addition, the proposed approach is sufficiently well scaled to increase the number of modules. In particular, the addition of new segments to robot design in the experiments performed has a little impact on the effectiveness of learning because it does not change the number of common rules for modules. However, it should be noted that the effectiveness of the proposed approach depends to a large extent on the number of similar modules in the robot structure. With the decrease of similar modules number, the advantages of using general rules are lost. From a practical point of view, the experimental studies on the basis of snake-like and multiped models of robots, as well as the trunk-like manipulator, have shown that the proposed approach is sufficiently effective and can be used to control complex modular systems with a large number of degrees of freedom.

9. Acknowledgments

This work is executed at financial support of RFBR Grant No.14-07-00386 and No.15-07-03410.

References

[1] Yim M.H., Duff D.G., Roufas K.D. Modular reconfigurable robots, an approach to urban search and rescue // 1st International Workshop on Human Welfare Robotics Systems (HWRS2000). – 2000. – pp. 19-20.

[2] Stoy K., Brandt D., Christensen D.J. Self-Reconfigurable robots: an introduction // Intellegent robotics and autonomous agents series. – MIT Press, 2010. – 216 p.

[3] Bongard J.C. Evolutionary Robotics // Communications of the ACM. – 2013. – Vol. 56. – No. 8. – pp. 74-83.

[4] Kamimura A., Kurokawa H., Yoshida E., Tomita K., Murata S., Kokaji S. Automatic locomotion pattern generation for modular robots // Proceedings of 2003 IEEE International Conference on Robotics and Automation. – 2003. – pp. 714-720.

[5] Ito K., Matsuno F. Control of hyper-redundant robot using QDSEGA // Proceedings of the 41st SICE Annual Conference (2002). – 2002. – V. 3. – pp. 1499-1504.

[6] Marbach D., Ijspeert A.J. Co-evolution of configuration and control for homogenous modular robots // Proceedings of the eighth conference on Intelligent Autonomous Systems (IAS8). – IOS Press, 2004. – pp. 712-719.

[7] Daoxiong Gong, Jie Yan, Guoyu Zuo. A Review of Gait Optimization Based on Evolutionary Computation // Applied Computational Intelligence and Soft Computing. – 2010. – vol. 2010. – Article ID 413179. – 12 p.

[8] Valsalam v. K. Miikkulainen R. Modular neuroevolution for multilegged locomotion//in Proceedings of GECCO. 2008. -pp. 265-272.

[9] Tanev I., Ray T., Buller A. Automated evolutionary design, robustness and adaptation of sidewinding locomotion for simulated snake-like robot//IEEE Transactions on robotics. -V. 21. -N. 4. -August 2005. -pp. 632-645.

[10] Mataric M., Cliff D. The challenge in evolving controllers for physical robot/robotics and autonomous systems. -October 1996. -19 (1). -pp. 67-83.

[11] Demin A.V. Logical model of the adaptive Control system based on functional systems theory//Young Scientist USA. Applied Science. -Auburn, Washington, 2014. -pp. 113-118.

[12] Demin A.V The model of the adaptive control system and its application for a virtual robot locomotion control //Molodoy Uchyony (Young scientist) - 2012, - No.11 (46) - pp. 114-119.

[13] Demin A.V. , Vityaev E.E. Logical model of the adaptive control system //Neuroinformatics. 2008, - vol. 3. – No. 1, - pp. 79-107.

[14] Demin A. V. Learning locomotion control system for 3D multiped robot model.//Molodoy Uchyony (Young scientist). - 2015, - No.19 (99), pp. 74-78.

[15] Demin A.V Teaching of locomotion ways of the snake-like robot virtual model//Molodoy Uchyony (Young scientist) 2014, - No.19 (78) - pp. 147-150.

[16] Smith R. Open Dynamics engine. Url: http://ode.org/.

[17] Evgenii Vityaev The logic of prediction // Proceedings of the 9th Asian Logic Conference (August 16-19, Novosibirsk, Russia), World Scientific Publishers, 2006, pp.263-276.

[18] Demin A.V., Vityaev E.E. Learning in a virtual model of the C. elegans nematode for locomotion and chemotaxis // Biologically Inspired Cognitive Architectures (2014). – Elsevier, 2014. – V. 7. – pp. 9-14.

[19] Demin A.V Learning control model of chemotaxis for C.Elegans nematode. // Neiroinformatics - 2013, vol. 7, No. 1, pp. 29-41