ISSN 1392 – 1215 ELEKTRONICS AND ELECTRICAL ENGINEERING
ELEKTRONIKA IR ELEKTROTECHNIKA
2006. Nr.4(68)

T125 AUTOMATION, ROBOTICS
AUTOMATIZAVIMAS, ROBOTOTECHNIKA

Coloured Petri Nets – Tool for Control Systems Learning

V. Baranauskas, K. Šarkauskas
Department of Control Engineering, Kaunas University of Technology,
Studentų 48, LT-51367 Kaunas, Lithuania; tel. +370 37 300290; e-mail: virgisbar@yahoo.com,
S. Bartkevičius
Department of Theoretical Electric Engineering, Kaunas University of Technology,
Studentų 48, Lt-51367 Kaunas, Lithuania; tel. +370 37 300253; e-mail: Stanislovas.Bartkevicius@ktu.lt

Introduction

To design and project control systems, especially if they are big or huge, nonlinear or stochastic, we need special software, which could simplify these processes, make them more comprehended and could accelerate the work. These types of system are often used, so design tools and software becomes more important. To project large or huge systems it is better to use intellectual methods, which are able to gather data, to distribute data during system work and could learn. Such modelling methods or ways are multi-agent systems and artificial neural networks [2, 4, 9].

Multi-agent systems are problem oriented, communicative, flexible and autonomous. It means that agents, acting in the system communicates between each other, they are flexible and they are not dependent from each other [2, 3].

Artificial neural network is understood as huge nonlinear system, decomposited from many simple nonlinear archive elements, named neurons. Artificial neural network distinguish adaptive control, optimization, pattern classification, pattern or signal filtration, pattern filling [4, 9].

These both systems are used for other system training, but it can‘t be efectivelly used in control systems.

Authors of the work suggest the use of coloured Petri nets, which can be used to design competitive process for learning control systems modelling. Traditionally talking about Petri nets, two points are accented – possibility to find and solve competition and investigate such Petri net characteristics as safeness, boundness, conservation, liveness, reach ability, coverability and etc. Besides, ability to model parallel processes and detect their competition are most important in most cases. There are many works, described how to use Petri nets to model control systems, but there are no all-around tools. So more and more often we find tasks, systems, in which Petri nets are not enough well known, which can’t be used for training, because there are no possibilities to remember solution ways. If the control system is very complex, it is better to simplify their models. Authors of the work suggest to use global variables, which simplify the structures of models and give new patterning possibilities in coloured Petri nets, which are aimed to model control systems and most important – it enables to use global variables for training of control systems [6, 8].

The comparable analysis, oriented to training problems is done for determining why coloured Petri networks with global variables is better than multi-agent systems and neural networks.

Multi-agent systems

Multi-agents can be used for modelling of learning systems. Often agent is described like computer program, acting in a narrow specific environment, which forms agent output actions using stored information, data, so that wanted goals could be reached [2].

Term agent is tight-knit associated with such attributes:

- *Problem oriented* – agents’ actions are pointed to solve narrow tasks.
- *Communicability* – agents must have real time connection with environment, by using sensors and actuators. At this point agents are different from classic systems, because most of them work in „off-line” mode.
- *Flexibility* – agents are dependent on present situation. Agent must pick best solution form many possibilities for problem solving and implement these solutions in time-varying dynamic environment.
- *Automomility* – agent must behave without direct human or other agent intervention. Agent must control his actions and internal state [2].

For difficult problems solving some agents could be joined together and form multi-agent system (Muli-Agent Systems, MAS). Main characteristics of these systems are:

- System is compiled form several agents, each of them controls only one or several items of information about object portions,
- There is no global control system,
- Information about system state is decentralized, calculating is executed asynchronously [2, 3].

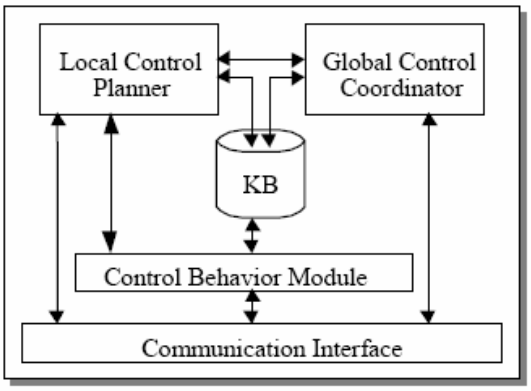

Fig. 1. Internal structure model of control agent

The Control Behaviour Module, Local Control Planner and Global Control Coordinator can make decisions within a partial hierarchical structure using their respective knowledge base views. The Control Behaviour Module responds to external inputs in a reactive manner to satisfy hard real-time control constrains and avoid explicit replanning. The Local Control Planner can devise plans and schedules for specific local goals, and attempts to realize soft real-time activities appropriate to the current controlled environment. When a situation exceeds the problem-solving capabilities of the Local Control Planner, control is shifted to the Global Control Coordinator. The Global Control Coordinator specifies long-term plans for agent behaviour when negotiating with other agents with respect to global control goals.

An agent learning is done through separate agent communication with other agents and their cooperation. Agents in the system exchange fixed information with each other and by that way themselves control their state. If one agent from a system done his own work, others agents in the system gets information, that the specific work is already done. Such agent communication and learning is very practical in manufacturing process, where moving robots are involved. In that way, each robot gets information about other robots movement directions and their coordinates. So attendant robots do not have to contact with each other, because they will exactly know where each of them are. In the further system work, if any former situation will repeat, robots will beforehand know what exactly to do. By these attributes, multi-agent system has main learning system elements.

There are several methods for projecting multi-agent systems. Best known of them is Gaia methodology.

Wooldridge, Jennings and Kinny conceived the Gaia method on basis of its experiences of many years of their work. Gaia is based on the abstraction of an agent system as organization a tree model, consisting of different interacting roles. System is divided in two parts: analysis and realization phases. During the analysis process, first, the individual roles in the system are identified and the interactions between the seen roles are specified. The description of a role contains the task of the role, its rights, activities and used minutes. In the specification of the task of a role, become the possible sequences and connections of all activities, as well as safety requirements, which must be kept. Activities are internal operational sequence of a role. Minutes are interaction mechanisms, with which dependence and relations between different roles are described. In the design phase, roles are combined into agent types. Further, we determinable possible communication paths between agent types and all services, which depend on each role of a type of agent specified. As result - a system project is presented, which is not enough however for an implementation and still must be further refined by other methods. During realization phase, agent and organization models are organized. Gaia is easily understandable and therefore is suitable for those who start up into the agent-oriented software development. A restriction from Gaia is that the relations between agent types are fixed and cannot change dynamically.

A multi-agent system has defects too. They are:

- While reading literature it is recondite, why for problem solving it is necessary to use multi-agent systems,
- by using multi-agent systems projectors attempt to reach general problem solutions, which could be used for all cases,
- Multi-agent systems are not standardized, so it is difficult to analyze and compare systems between each other [3].

Artificial neural networks

Artificial neural network is understood as huge nonlinear system, decomposited from many simple nonlinear archive elements, named neurons.

Main functions of artificial neural networks are:

- Modelling and calculating
- Adaptive control
- Optimization
- Pattern classification
- Pattern or signal filtration
- Pattern filling.

Modelling and calculating: Entrance and exit data of the system is used for autonomous and adaptive training, which is the way how neural nets models are receivable. So predictive or calculated system exit (or mode) could be made-up from well trained or adaptive neural network. In neural network technology, this function is used for patterns marking, approximations of the functions and in the associative memory [4, 9].

Adaptive control: Neural network could be use as a adaptive controller. Given that inaccurate environment of the system, good adaptive control productivity could be reached (mostly for nonlinear systems with unknown mathematical model).

Optimization: dynamical (with recursive reaction) neural nets can present almost optional problem solution. Pattern classification: well trained neural network could be used for identifying typical class from given packs patterns in the system entrance. This function is used during data pressure, feature exclusions, signal coding and other.

Pattern or signal classification: When in neural network entrances signals with noise are present (or distorted patterns), signals or patterns without noise or with lesser distortions could be obtained. This function is analogical to mathematical filtering models.

Pattern filling: Neural network can create complete pattern although incomplete pattern is given in the entrance.

Classic artificial neural network consists from entrance buffer, inside (hidden) layers and exit buffer.

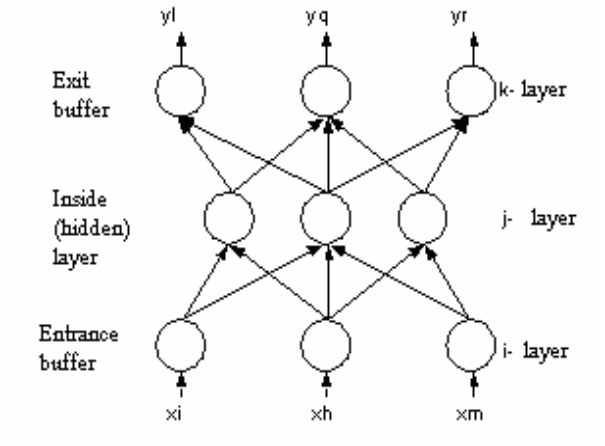

Fig. 2. Classic structure of neural network

Entrance buffer is buffer, which gives data for network. This entrance layer is not neural calculating layer, because nodes don‘t have weights and activating functions. Upper exit layer pictures response to given entrances. Inside layer is named hidden layer, because it doesn‘t have direct link to outside. These layers are typically named i-layer, j-layer and k-layer.

If you use suitable number of nodes in each hidden layer, using neural networks you could approximate very complex nonlinear dependences. Neural networks don‘t have strictly fixed functional form and for example number of neurons in hidden layer, during modelling process is often alternate. During modelling, particular temporary neural network structure is selected and network weight coefficients are identified during network training process. Network structure could be changed, depending on modelling results. Often received model is better than polynomial or harmonium functions models if you choose suitable neural network structure [9].

An artificial neural network, in article author’s oppinion, is not recommended to use for training manufacture line. Solution is motivated by fact, that when neural network is training, some conflict situations could arise, which is not acceptable during manufacture process.

Learning coloured Petri nets

Coloured Petri nets (CPN) – it is expanded Petri nets (PN), in which tokens are differentiated by colours, transitions and a connection identifies their migration in the net during modification.

The notation of coloured PN is much concise than classical nets. Many relapses could be evaded, which are inherent to classic PN.

SML (Standard Meta Language) is used as base for description and calculating. It is modified and stored for effectively work. Control systems are real objects, which exist near us and functional in the same time and space. So time is one of the main accents in this software, which solves functionality of the net. This enables us to project control systems in the same real categories, like-for-like it makes all conditions for creation control programs in various control systems [1, 8].

Centaurus Coloured is specialized software. You can model systems, which have algorithmic part, which is described using elements of CPN and analogical part, which is described by structural scheme using this software.

There are three elements of PN – position, transition and chord, and new element, named “process” in this software. Large industrial systems could be modelled using such model structure.

One of the ways of using modelling results is creation of control program or its algorithm, or even automatic generation for purpose to install to real control system. The question is, why can‘t we install already trained control program to machine? There were no such possibilities in CPN, because there only systems existed with control program pended. During development of control program, system can function due to provided algorithm. The training becomes very effective, if some choices in system work are possible. In usual systems, choice is made by projector, a priori choosing proper, but not always optimal version. It is proved, that if you have complicated system and asynchronous processes in here, the effectivity of system work rises if control algorithms, accepting implemented solutions, not only due to present task, but appreciating next task, are used [10]. In such control system, usual PN, that problem can’t be solved, so the global variables are used in Centaurus coloured software. The definite rules are established for applying global variables in CPN, so it couldn‘t awake any ambiguities in modeling system.

For usage of global variables in CPN, the following rules are proposed:

1. global variables can not be used in constructors;
2. new values may be assigned to global variables only through finish expressions*;*
3. global variables can obtain new values only at the end a simulation step;

Each step of simulation of a PN with global variables consists of stages:

1. verification of conditions of firing of all transitions;
2. evaluation of expressions of output chords if fired transitions and placement of calculated multi-sets into output places;

3. calculation and assignment of new values to global variables.

Values of global variables do not change along a simulation step, but could be changed just before start of another one.

Global variables enable to remember modeling conditions, so you can choose the best solution from several variants, that means that you can create all possibilities for system training.

The other way for using model results for training system is the current system perfection in projection phase. As example let‘s analyze machine room with serving robots. Depots are needed for planchets and already made components. While the system is not large (one robot and few machines), robot actions could be programmed by a human. This means that all possible combinations are inspected manually; little usable depots are not used, robot and machines work is distributed that the downtime would be as possibly less. This type of manual modeling is very complex or impossible, requires a lot of time and resources, if the number of machines, details and planchets is rising up. The rational way is to create training system [5, 6, 7].

There are two possible training ways: 1. when training of the system is done during manufacture process, 2. when the training process is done in the software, and then, trained program is loaded into real system.

In the first way, during manufacture process, machines and robots must react to emerging difficulties; learn how to avoid robots confrontation by themselves. If any problem with machines or robots arises, all manufacturing process falls into disarray. While the system coordinates all actions of devices, new problems could be evoked. System work is perturbed if one of machine is damaged and it is exported for fixing. At this case, some free space is left in the room. Then robots must recalculate coordinates of machine places and eliminate exported machines. As we see, that way of system training is not rational, baffling manufacture and even menaces hardware.

In the second way, system is trained according to model, while manufacture cycle is imitable. The whole control model is generated by Centaurus Coloured software. The training system must function like that: all possible variants of situation must be checked, fixing best variants. At this time problem emerges: how and where the best training results should be kept. So there must be global variables, which could be accessed any time in a model. During modeling, training process is realized during imitation of a given system work model. This means, that in Centaurus Coloured software, system work process is simulated, received results are fixed in model database. When it is done, due to machines load and usage of robots, the best solution of system work is chosen according to these requirements. Such training schedule could be used doubly: in project phase, when you need to pick proper machines positions and functional phase, when the trained system model must be written into PLC’s.

Let’s illustrate it with example. Suppose we have three machines system, which is served by one robot. Robot brings three kinds of planchets and picks production from them. Each planchet and component is transported to different depot. So we have three depots for planchets and components on each side of the manufacturing line. Such system model, modeled in Centaurus Coloured software, is given in Fig. 3.

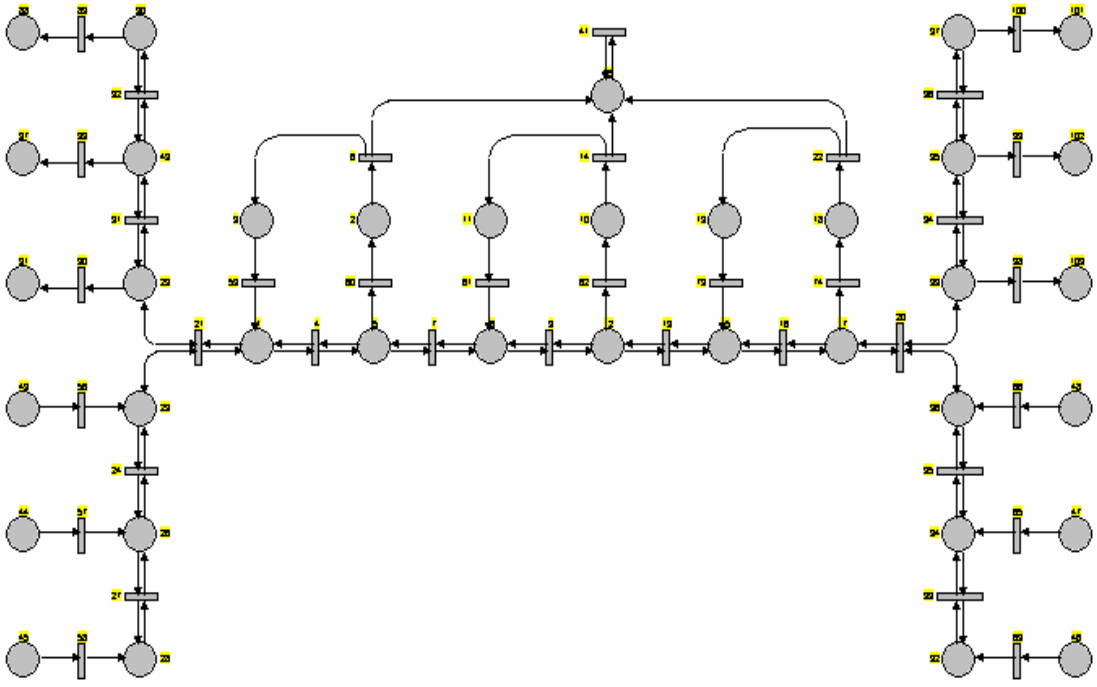

Fig. 3. Three machines system, which is served by robot, with three products and planchets depots

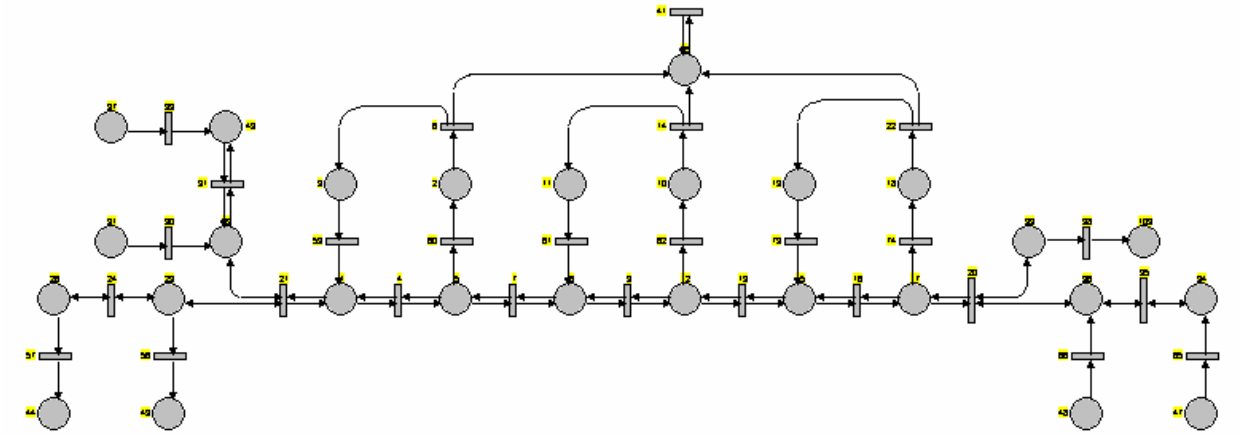

Fig. 4. Reoriented depots-manipulator line model in Centaurus Coloured software, when training results are evaluated

While looking to given model in Fig. 3, question arises: is every depot used effectively and does the system work without outage and is not overloaded? If you want to answer this question, you should calculate how many components and planchets are transported to every depot. While calculating robot movement, we consider that when robot goes to depot to pick up planchet, it could carry produced component to depot too.

When all possible variants are investigated, conclusion is made that in the right side of the scheme only three depots are used, and in the left side – only four from six. It is given in Fig. 4.

Training process is very similar to optimization process in other systems. The situation is generated during optimization process, their optimized parameters are changed and criterion value is determined again. If the value of criterion mutate in desirable direction, further mutation in the same direction is beeing done. The essence is that the system is simulating again and again, until desirable results are reached. At this point, training process becomes like that: first situation is generated and it is trying to find optional solution from many possible solutions, situation parameters are verified and concrete results are written into database. Later, the next situation is generated, all possible fulfilments of current situation are checked and the best result is fixed and so on, while all possible situations are checked. Exactly here, global variables are needed in Petri nets. So then results could be accumulated in model and it could be reached any time, from any place during design process.

The creation of global variables in Centaurus Coloured software and its usage for training process is the first step for adaptation to use it for designing of training systems.

Conclusions

1. Multi-agent systems are suitable for solving learning problems and they can be effectively used for learning during manufacture process.

2. In neural networks while training them, some conflict situations could be originated. It could upstart for probabilistic solution logic, which can evoke unfavourable results during manufacturing process.

3. The rational way to solve training problem is to use CPN. Global variables are needed if you want to implement this solution. System training is done in „off-line" mode.

References

1. **Bartkevičius S. K., Mačerauskas V., Šarkauskas K.** Spalvotųjų Petri tinklų taikymas valdymo sistemoms modeliuoti // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2003. – Nr. 4(16). – P. 7–11.
2. **Baranauskas V.** Multi-agentinės valdymo technologijos taikymas automatizavimo sistemose: Magistro darbas. – Kaunas, 2005.
3. **Simutis R.** Multiagentų metodikos taikymas procesų valdymo sistemose // Automatika ir valdymo technologijos: tarptautinės konferencijos medžiaga. – 2002. – P. 12–18.
4. **Simutis R., Levišauskas D., Stankevičius G.** Procesų ir sistemų modeliavimas. – Kaunas: Technologija, 1999. – P. 86–88.
5. **Wermter S., Elshaw M.** Learning robot actions based on self-organising langue memory // Neural Networks.– Volume 16, Issues 5–6, June–July 2003. – P. 691–699.
6. **Hirasawa K., Ohbayashi M., Sakai S., Hu J.** Learning Petri network and its application to nonlinear system control // IEEE Transactions on systems, man and cybernetics. – Part B: Cybernetics.– Vol. 28, No. 6, December 1998. – P. 781–789.
7. **Provost J., Kuipers B. J., Miikkulainen R.** Self-organizing distinctive-state abstraction for learning robot navigation // Artificial Intelligence Lab, The University of Texas at Austin, 1 university Station C0500, Austin TX 78712 USA. July 2005. – P. 1–9.
8. **Kragnys R., Bartkevičius S.** Hibridinių valdymo sistemų modeliavimo Petri tinklais ypatumai // Elektronika ir elektrotechnika – Kaunas: Technologija, 2005. – Nr. 6(62). – P.82–87.
9. **Ripley B. D.** Pattern recognition and neural networks. // Textbook, Cambridge University, UK, ISBN 0 521 46086 7 1996. – P. 1–416
10. **Bartkevičius S., Daunoras J., Šarkauskas K.** Lanksčios linijos valdymo optimizavimas // Elektronika ir elektrotechnika: Mokslo darbai. – Kaunas: Technologija, 2005. – Nr. 1(57). – P. 56–61.

Presented for publication 2006 03 01

V. Baranauskas, S. Bartkevičius, K. Šarkauskas. Coloured Petri Nets – Tool for Control Systems Learning // Electronics and Electrical Engineering. – Kaunas: Technologija, 2006. – No. 4(68). – P. 41–46.

Control systems become increasingly more complex, and sometimes the sequence of technological process so frequently changes, that the creation of the programs of control becomes problematic. One of the ways is to use learning systems. For purposes of learning the principles of multi-agent systems or the methods of artificial neural nets are used. The principles of multi-agent systems are most acceptable in the real devices, but the process of learning creates failures and damages the technological process. Artificial neural nets solve the same problem by probabilistic methods, so that during the learning the same problems appear. For purposes of learning coloured Petri nets are well suited if we use the global variables in them, which create the possibility to form, accumulate and preserve data of instruction. The model of system in the coloured Petri nets consists of two large parts; these are the model of the program of control and the model of technological process. The learning of system is conducted on the model and the already trained program of control is transferred to the control device. Ill. 4, bibl. 10 (in English; summaries in English, Russian and Lithuanian).

В. Баранаускас, С. Барткевичюс, К. Шаркаускас. Цветные Петри сети – устройство для обучения систем управления // Электроника и электротехника. – Каунас: Технология, 2006. – № 4(68). – С. 41–46.

Системы управления становятся все сложнее, а иногда последовательность технологического процесса так часто меняется, что становится проблемным создание программ управления. Один из путей – это применение для таких целей обучающиеся системы. Для целей обучения используются принципы мултиагентных систем или методы искусственных неуросетей. Принципы мултиагентных систем наиболее приемлемы в реальных устройствах, но процесс обучения создает сбои и повреждения технологического процесса. Искусственные неуросети ту же проблему решают вероятностными методами, так что во время обучения возникают те же проблемы. Для целей обучения хорошо подходят цветные Петри сети, если в них использовать глобальные переменные, которые создают возможность формировать, накапливать и сохранять данные обучения. Модель системы в цветных Петри сетях, состоит из двух крупных частей, это модель программы управления и модель технологического процесса. Обучение системы проводится на модели, а уже обученная программа управления переносится на управляющее устройство. Ил. 4, библ.10 (на английском языке; рефераты на английском, русском и литовском яз.).

V. Baranauskas, S. Bartkevičius, K. Šarkauskas. Spalvoti Petri tinklai – valdymo sistemų savimokos priemonė // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2006. – Nr. 4(68). – P. 41–46.

Valdymo sistemos, kurios tampa vis sudėtingesnės ar kuriose kartkarčiais reikia keisti technologinio proceso eigą, kelia problemų sudarant valdymo programas. Vienas iš būdų šiai problemai spręsti yra apsimokančių sistemų taikymas. Savimokos tikslams taikomi daugiaagentės sistemos principai ir dirbtinių neuroninių tinklų metodai, tačiau yra nemaža neišspęstų problemų, trukdančių taikyti juos gamybinėms valdymo sistemoms mokyti. Daugiaagentės sistemos principai geriau tinka realiems įrenginiams, tačiau tokios sistemos savimoka susijusi su gamybos technologijos trikdžiais ir pažeidimais mokymo metu. Dirbtiniai neuroniniai tinklai tą pačią problemą sprendžia tikimybiniais principais, dėl to mokymo metu gali kilti tų pačių problemų. Valdymo sistemoms mokyti labai gerai tinka spalvoti Petri tinklai, jei juose naudojami globalūs kintamieji, kurie leidžia formuoti, kaupti ir išsaugoti apmokymo duomenis. Sistemos modelis spalvotuose Petri tinkluose susideda iš dviejų stambių dalių –valdymo programos modelio ir technologinio proceso modelio. Sistema apmokoma modelyje, po to jau išmokyta valdymo programa įvedama į valdymo įtaisą. Il. 4, bibl. 10 (anglų kalba, santraukos anglų, rusų ir lietuvių k.).

DOI: 10.5755/j02.eie.10647