ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2007. Nr. 4(76)
ELEKTRONIKA IR ELEKTROTECHNIKA

AUTOMATION, ROBOTICS
T 125
AUTOMATIZAVIMAS, ROBOTECHNIKA

Training Possibilities in CPN

V. Baranauskas, K. Šarkauskas
Department of Control Engineering, Kaunas University of Technology,
Studentų st. 48-319, LT-51367 Kaunas, Lithuania, phone +370 37 300292, e-mail: virgisbar@yahoo.com,
Studentų st. 48-107, LT-51367 Kaunas, Lithuania, phone +370 61025585, e-mail: Kastytis.Sarkauskas@ktu.lt

S. Bartkevičius
Department of Theoretical Electric Engineering, Kaunas University of Technology,
Studentų st. 48-230, LT-51367 Kaunas, Lithuania, phone +370 37 300253, e-mail: Stanislovas.Bartkevicius@ktu.lt

Introduction

Flexible production systems must harmonize asynchronous processes [1]. Such systems become even more complicated, with respect to control, when mobile robots are used. Single robot is an autonomous system self-organizing solution of a task, but this task often is a part of complicated production process. Central control system is required to formulate and distribute tasks for robots.

This paper deals with control of flexible production system with mobile transportation robots. The main task of these robots is to serve production centers and storages. Central control system formulates exact task for a robot.

A robot, realizing this task, must determine a movement trace, speed *etc*. The trace must be determined avoiding collisions with obstacles, rating clearances of the tote and robot itself. The obstacles may be static, not moving, and dynamic. It seams possible to treat dynamic obstacles as "disturbances" and problem to avoid them is to solve dynamically, so movement traces depends only on presence of static obstacles. So planned traces are "static" – depends only of start and finish points and static obstacles. These traces may repeat for different robots, so it is useful to remember them as an "experience". Solution of conflicts with dynamic obstacles is not the purpose of this paper.

Sometimes artificial neural networks are used for modeling of control systems, but these models of real systems are very large and complicated for analysis. This problem caused attempts to lookup alternatives, Petri nets, for example [2].

Authors of the work suggests to use 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. Possibility to simulate parallel in time and asynchronous processes seams to be the most important property simulating before mentioned systems. There are many works, described how to use Petri nets to model control systems [3, 4], but there are no all-round tools. So more and more often, we find tasks, systems, in which are not enough well known Petri nets, 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 suggests to use global variables [5, 6], which simplifies structures of models and gives new patterning possibilities, in cololoured Petri nets, which are aimed to model control systems and most important – it enables to use global variables for training control systems [1, 7]. The system for teaching of robots is proposed.

Requirements and constraints

Let's suppose, that system to simulate has star like structure. Robots can communicate with direct numerical control system (Fig. 1.), but communication between robots is minimal – any robot can sense only presence of another robot near.

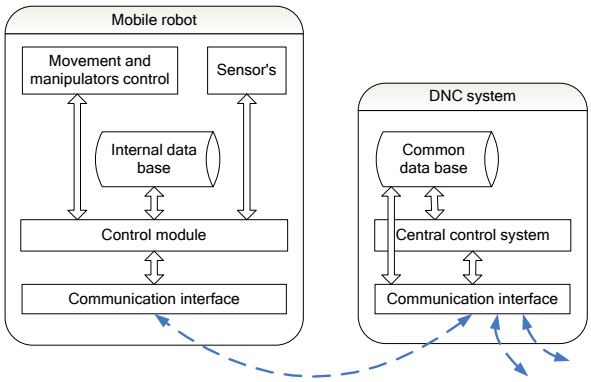

Fig. 1. Structures of a mobile robot and direct numerical control system

The mobile robot can make decisions in an "intelligent" manner; a relevant knowledge base is a core component of it. The Direct numerical control (DNC) system can make decisions within a partial hierarchical structure using their respective knowledge base views. The DNC system responds to external inputs in a reactive manner to satisfy hard real-time control constrains and avoid explicit replanning. The Control module of mobile robot 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 Control module of mobile robot, control is shifted to the DNC system. The DNC system specifies long-term plans for agent behaviour when negotiating with other robots with respect to global control goals.

Such robot communication, for teaching or learning, is very practical in manufacturing process, where moving robots are involved. In that way, each robot gets training information about other robots moving directions and their coordinates. So attendant robots can't contact with each other, but its changed information thought CNC system, and new training or learning information will be saved into common data base. In the further system work, if any former situation will repeat, robots will beforehand know what exactly to do. The communication interface provides communication and negotiation protocols for data message communication within various levels of time constraint.

Training in colored Petri nets

The asynchronous system discussed above is handy to simulate using colored Petri nets (CPN) with syntax extension allowing global variables [5, 6]. Presence of global variables makes system able to remember data, so provide databases. Protocols of communication between various components are not essential thing simulating behavior of a system. The main purpose is to find the shortest traces of robots and avoid collisions with obstacles.

Authors has written simulation tool *CENTAURUS 2D*. This tool allows visualization of movement of objects on a plane. Movement and interaction of various objects, static and dynamic, describes by means of colored Petri nets. Graphic subsystem shows shapes of moving robots and obstacles, marks collisions.

The new default color is necessary for interface with graphic subsystem. Any moving object is associated with a "region" – a polygon. This object is of color REGION (notation in SML language):

```
Color Region = record Hidden: bool *
                      VetoIn:  bool *
                      Color:   int *
                      Angle:   real *
                        PtX:   list real *
                        PtY:   list real;
```

The field *Hidden* defines, if a region is visible or hidden, *VetoIn* indicates that interior of a region is forbidden or not for other regions, *Color* defines a color of edges, *Angle* is an angle of which a region must be rotated when drawn, *PtX* and *PtY* – lists of coordinates of vertexes.

When a robot moves, an associated region moves in graphic window accordingly. All associated regions are declared as global variables in declaration part of Petri model. Coordinates are relative and define vertexes versus an imaginary "zero point". The graphic subsystem shifts vertexes by real coordinates of a robot in each step of simulation.

Static objects, obstacles, are regions in graphic window too. Graphic subsystem detects intercrossing of regions and breaks simulation, if any happens.

A fragment of a production system with mobile transport robots is presented in Fig. 2.

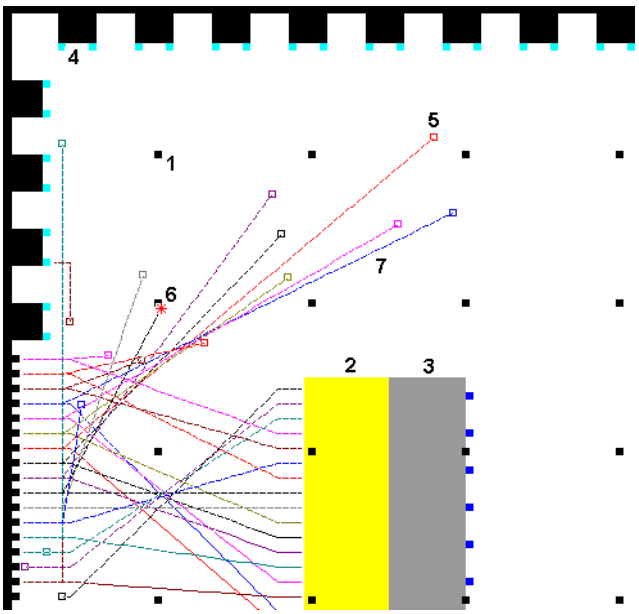

Fig. 2. A fragment of production system with mobile transportation robots: 1 – static object (a column), 2 – static object (zone forbidden to robots), 3 – static object (zone of assembling of a product), 4 – static object (production center), 5 – dynamic object (mobile robot), 6 – collision of a mobile robot with a column, 7 – trace of a mobile robot

Principles of representation of system (Fig. 1) and simulation are described above.

Any trace of a robot has start and finish points declared as components of lists of points (coordinates *x* and *y*):

```
val RuosIn3=[(8.5,24.5),(3.5,24.5),(2.5,24.5)];
val RuosOut3=[(2.5,24.5),(7.5,24.5)].
```

The declaration represents a third port of store. Three points *RuosIn3* represent input into the store fragment, because robot must reduce its speed moving from point to point, fix its position and stop at the point *(2.5,24.5)*. Two points *RuosOut3* describe start from this port – distance between these points is an acceleration interval. Descriptions of other ports are analogous.

Any trace of a robot starts and finishes at ports, so it is a list of points starting with two output points of a port ant finishing by three input points of an another one. All intermediate points between these two groups represent "corners" of a trace and aim to avoid static obstacles and ensure possible shortest way.

If a robot has finished movement from one port to another, this experience is useful not only for it. This is useful for any robot moving by the same trace. So there is necessity, to make a model of a system able to remember successful traces and interchange with them between robots. The database of traces is necessary. Global variables – lists of points can serve for this purpose. Hazards uprising if global variables are used and measures to avoid them are discussed in [5, 6]. Another necessary extension of colored Petri nets is possibility to fix this dynamically obtained database in declarations of a system model. The new default function serves for this purpose:

FixDecl(GlobalVarName): bool.

This function renews declaration of global variable pointed by *GlobalVarName* according its actual content. Result of this function is *true*, if restore is successful.

Presence of obstacles makes traces of robots not straight. The graphic subsystem of CENTAURUS 2D enables not only fix collisions between objects, but makes possible to shift trace, insert a new corner point, in aim to avoid this conflict.

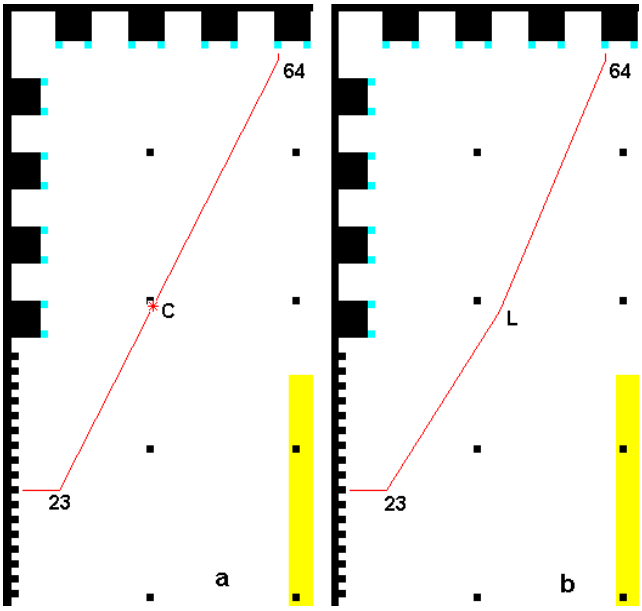

Fig. 3. Training of a mobile robot in graphic subsystem of program package CENTAURUS 2D. 23 – a start point of a trace of mobile robot, 64 – a finish point of a trace, C – point of collision, L – new corner point to avoid obstacle (column)

The training of the robot after collision (Fig. 3) is graphical movement of a point of the trace pointed by mouse marker. The graphic subsystem and Petri net interchanges information with Petri net via two functions:

MakeTrace(RegName: string): way;

and

FixTrace(Trc: way, RegName: string): bool.

Type (color) *way* represents list of points described before. *RegName* – name of a region associated with appropriate robot. Only headers of these functions are default. A user writes bodies of them. Function *MakeTrace* exports a trace associated with a robot to the graphic subsystem and *FixTrace* returns corrected trace to Petri model. Such interface does not constrict a user in usage of traces.

Fig. 4 shows traces left of successfully trained robots of the simulated production system.

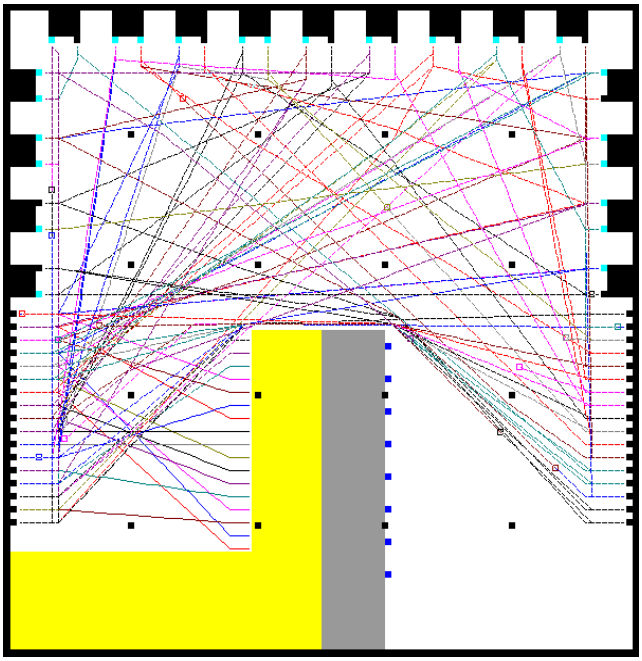

Fig. 4. Traces of trained mobile robots in the production system simulated by package CENTAURUS 2D. Filled areas are static objects or zones, where robots could not access, other lines – the ways of robot movement

Training of a mobile robot seems to be quite simple thing, but a problem arises when number of terminal points of possible traces reaches tenths. It seems to be reasonable to organize two-way traces – the traces, from *a* to *b* and from *b* to *a*, coincide only in terminal points. Such two-way trace allows moving two robots in opposite directions without collision. The simulated system has $r = 17$ initial positions of robots, $s_p = 17$ stores of planchets, $s_d = 17$ stores of made details, $c = 17$ production centers and $a = 8$ entry points of assembling line. Total number of possible different traces is

$$n_t = s_p \cdot (r + c*2 + s_d + a) + s_d(c + (a + s_d)\cdot 2) + \\ + 2s_d a = 2703. \quad (1)$$

On common, the number of possible traces increases, if all traces between whichever pair of terminal points are meaningful. If number of terminal points $t = 17$, total number of traces is

$$n_c = \frac{t!}{(t-2)!} = 4556. \quad (2)$$

These numbers shows, that manual training of robots make sense only for cases with small number of terminal points. An automatic training system is required for systems that are more complicated. The package CENTAURUS 2D has presumptions to do it in further development of the package.

Conclusions

1. Program package allowing simulation, based on color Petri nets, of movement of mobile objects and detect collisions proposed.
2. Manual training of mobile robots using graphical subsystem and providing database of traces realized in the package.
3. The realized default procedures and interface with graphic subsystem probably makes presumptions for learning of mobile robots.

References

1. **Bartkevičius S., Daunoras J., Šarkauskas K.** Lanksčios linijos valdymo optimizavimas // Elektronika ir elektrotechnika: Mokslo darbai.- Kaunas: Technologija, 2005. – Nr. 1(57). – P. 56–61.
2. **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.
3. **Simutis R.** Multi-agentų 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. **Bartkevičius S., Kragnys R., Šarkauskas K.**. Global Variables in Colored Petri Nets // Elektronika ir elektrotechnika: Mokslo darbai. – Kaunas: Technologija, 2006. – Nr. 5(69). – P. 49–52.
6. **Baranauskas V., Bartkevičius S., Kragnys R., Šarkauskas K.**. Modeling Control Systems with Coloured Petri Nets Using Global Variables // Electrical and control technologies-2006: Proceedings of international conference. – Kaunas: Technologija, 2006. – P. 306–309.
7. **Kragnys R., Bartkevičius S.** Hibridinių valdymo sistemų modeliavimo Petri tinklais ypatumai // Elektronika ir elektrotechnika – Kaunas: Technologija, 2005. – Nr. 6(62). – P. 82–87.

Presented for publication 2007.03.01

V. Baranauskas, K. Šarkauskas., S. Bartkevičius. Training Possibilities in CPN // Electronics and Electrical Engineering. – Kaunas: Technologija, 2007. – No. 4(76). – P. 25–28.

Flexible production systems require usage of mobile transportation robots. Training of these robots is necessary to ensure effective and conflict less operation. The principles of multi-agent systems and artificial neural nets may be used in training processes, but some not solved yet problems make usage of these methods complicated. Colored Petri nets are well suited for purposes of training, if extension of usage of global variables made. Presence of these variables creates the possibility to accumulate data obtained during training and provide databases. The training of system is conducted on the model; collisions are resolved using graphical subsystem and database, accessible for all mobile robots, provided. The database of the already trained system of control can be transferred to the control device. Ill. 4, bibl. 7 (in English; summaries in English, Russian and Lithuanian).

В. Баранаускас, К. Шаркаускас, С. Барткевичюс. Возможности обучения в цветных Петри сетях // Электроника и электротехника. – Каунас: Технология, 2007. – № 4(76). – С. 25–28.

В гибких линиях часто меняется последовательность технологического процесса, так что для решения многих проблем желательно использование мобильных роботов. Одним из путей повышения эффективности функционирования является начальное обучение мобильных роботов. 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 подходит цветные Петри сети, неимеющие перечисленных недостатков, если в них использовать глобальные переменные, которые создают возможность формировать, накапливать и сохранять данные обучения. Обучение системы проводится на модели, конфликтные ситуации решаются на графической подсистеме программного пакета, а результаты обученная накапливаются в общей базе данных. Ил. 4, библ.7 (на английском языке; рефераты на английском, русском и литовском яз.).

V. Baranauskas, K. Šarkauskas, S. Bartkevičius. Mokymo spalvotuosiuose Petri tinkluose galimybės // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2007. – Nr. 4(76). – P. 25–28.

Lanksčiosiose linijose, kuriose technologinio proceso eigą kartkarčiais reikia keisti, yra naudojami mobilieji robotai. Kad valdymo sistemos efektyviai funkcionuotų, reikalingas pradinis mobiliųjų robotų apmokymas. Apmokymo tikslams naudojami multiagentų sistemos principai ir dirbtinių neuroninių tinklų metodai, tačiau yra nemaža neišspęstų problemų, kuriuos trukdo taikyti juos gamybinėms valdymo sistemoms mokyti. Valdymo sistemų tikslai yra aiškiai determinuoti, todėl abejotini ar dviprasmiški sprendimai neleistini. Valdymo sistemoms mokyti labai gerai tinka spalvotieji Petri tinklai, jei juose naudojami globalieji kintamieji, kurie įgalina formuoti, kaupti ir išsaugoti mokymo duomenis. Sistema mokoma modelyje, konfliktines situacijas sprendžiant grafiniame programinio paketo posistemyje, o mokymo rezultatai kaupiami bendroje visiems mobiliesiems robotams duomenų bazėje, kuri gali būti naudojama valdymo įrenginyje. Il. 4, bibl. 7 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).

DOI: 10.5755/j02.eie.10710