Implementing Obstacle Avoidance and Follower Behaviors on Koala Robots Using Numerical P Systems

Cristian Ioan Vasile[1], Ana Brânduşa Pavel[1], Ioan Dumitrache[1], and Jozef Kelemen[2]

[1] Department of Automatic Control and Systems Engineering
Politehnica University of Bucharest
Splaiul Independenţei 313, 060042 Bucharest, Romania
{cvasile, apavel, idumitrache}@ics.pub.ro
[2] Institute of Computer Science
Silesian University in Opava
kelemen@fpf.slu.cz

Summary. Membrane controllers have been developed using Numerical P Systems and their extension, Enzymatic Numerical P Systems, for controlling mobile robots like e-puck and Khepera III. In this paper we prove that membrane controllers can be easily adapted for other types of robotic platforms. Therefore, obstacle avoidance and follower behaviors were adapted for Koala robots. The membrane controllers for Koala robots have been tested on real and simulated platforms. Experimental results and performance analysis are presented.

1 Introduction

Numerical P systems represent a type of membrane systems introduced by Gh. Păun in [7]. The main difference of this computational model in comparison to other types of P systems [8] is that compartments contain numerical variables (instead of symbols) which evolve by means of programs (rules). A membrane system has a tree-like structure and computation takes place in parallel in all membranes. Using membrane computing paradigm for modeling robot controllers is a new approach that was discussed and analysed in several papers [2], [5], [1]. Numerical P Systems (NPS) and their extension, Enzymatic Numerical P Systems (ENPS), were successfully applied for modeling robot behaviors like obstacle avoidance, wall following, following another robot, localization [2], [5], [6]. The robot behaviors were tested on real and simulated two wheel differential robots: e-puck and Khepera III. These two types of robots are small educational robots with diameters between about 7 and 13 cm. In this paper, we propose robot controllers for following a leader and obstacle avoidance behaviors, which were tested on Koala

educational robots. Koala robot is a bigger robot, about 30x30 size, with different infrared sensor placement than e-puck and Khepera III. In this way we prove that membrane controllers can be easily adapted to any types of robots and are a scalable modeling tool for robotics applications. The membrane controllers for the follower behavior are modeled based on classical control laws (proportional controller) using membrane systems. The obstacle avoidance behavior is presented in detail in [5]. The membrane controllers' performance will be analyzed and presented in this paper.
In order to introduce the NPS and ENPS models and the mathematical notations, we further present formal definitions which were adapated from other papers. For instance numerical P systems are presented in detail in [7]. Their definition is the following:

$$\Pi = (m, H, \mu, (Var_1, Pr_1, Var_1(0)), \ldots, (Var_m, Pr_m, Var_m(0))) \tag{1}$$

where:

- m is the number of membranes used in the system, degree of Π; $m \geq 1$;
- H is an alphabet that contains m symbols (the labels of the membranes);
- μ is a membrane structure;
- Var_i is the set of variables from compartment i, and the initial values for these variables are $Var_i(0)$;
- Pr_i is the set of programs (rules) from compartment i. Programs process variables and have two components, a production function and a repartition protocol.
  The j-th program has the following form:

$$Pr_{j,i} = (F_{j,i}(x_{1,i}, ..., x_{k_i,i}), c_{j,1}|v_1 + ... + c_{j,n_i}|v_{n_i}) \tag{2}$$

  where:
  - $F_{j,i}(x_{1,i}, ..., x_{k_i,i})$ is the production function;
  - k_i represents the number of variables in membrane i;
  - $c_{j,1}|v_1 + ... + c_{j,n_i}|v_{n_i}$ is the repartition protocol;
  - n_i represents the number of variables contained in membrane i, plus the the number of variables contained in the parent membrane of i, plus the number of variables contained in the children membranes of i.

The variables $c_{j,1}, \ldots, c_{j,n_i}$ are natural numbers (they may be also 0, case in which it is omitted to write "$+0|x$") [7]. These coefficients specify the proportion of the current production distributed to each variable $v_1, ..., v_{n_i}$. Let,

$$C_{j,i} = \sum_{n=1}^{n_i} c_{j,n} \tag{3}$$

A program $Pr_{j,i}$ is executed as follows. At any time t, the function $F_{j,i}(x_{1,i}, ..., x_{k_i,i})$ is computed. The value:

$$q = \frac{F_{j,i}(x_{1,i}, ..., x_{k_i,i})}{C_{j,i}} \tag{4}$$

represents the "unitary portion" to be distributed to variables $v_1, \ldots, v_{n_i}$, according to coefficients $c_{j,1}, \ldots, c_{j,n_i}$ in order to obtain the values of these variables at time $t + 1$. Specifically, variable v_s which belongs to the repartition protocol of program j, will receive:

$$q * c_{j,s}, for 1 \leq s \leq n_i \tag{5}$$

The variables which receive new values from a rule must be contained within the current, the parent or a child membrane. If a variable belongs to membrane i, it can appear in the repartition protocol of the parent membrane of i and also in the repartition protocol of the child membranes of i. After applying all the rules, if a variable receives such "contributions" from several neighboring compartments, then they are added in order to produce the next value of the variable.

A production function which belongs to membrane i may depend only on some of the variables from membrane i. Those variables which appear in the production function become 0 after the execution of the program.

Deterministic NPS have only one rule per membrane ($card(Pr_i) = 1$) or must have a selection mechanism that can decide which rule to apply. The NPS model with multiple rules per membrane is a non-deterministic system. However, NPS are well suited for applications which involve numerical variables and require a deterministic behavior, such as control systems for mobile robots. Thus a selection mechanism for the active rules is defined in the extended model, enzymatic numerical P systems (ENPS), proposed [3].

ENPS is defined as a NPS with special enzyme-like variables which control the execution of the rules:

$$\Pi = (m, H, \mu, (Var_1, E_1, Pr_1, Var_1(0)), \ldots, (Var_m, Pr_m, E_m, Var_m(0))) \tag{6}$$

where:

- E_i is a set of enzyme variables from compartment i, $E_i \subset Var_i$
- Pr_i is the set of programs from compartment i. Programs have one of the two following forms:
  1. non-enzymatic form, which is exactly like the one from the standard NPS:

$$Pr_{j,i} = (F_{j,i}(x_{1,i}, \ldots, x_{k_i,i}), c_{j,1}|v_1 + ... + c_{j,n_i}|v_{n_i}) \tag{7}$$

  2. enzymatic form

$$Pr_{j,i} = (F_{j,i}(x_{1,i}, \ldots, x_{k_i,i}), e_{t,i}, c_{j,1}|v_1 + ... + c_{j,n_i}|v_{n_i}) \tag{8}$$

  where $e_{t,i} \in E_i$

There can be more than one active rule in a membrane or none. A rule is active if it is in the non-enzymatic form or if the associated enzyme has a greater value than one of the variables involved in the production function, in absolute value. All active rules in the membrane system are executed in parallel in one computational step. The enzymatic mechanism and the advantages of ENPS are detailed in [5], [1].

2 Behaviors on Koala robot

In this paper two behaviors are presented and tested on simulated and real Koala robots: obstacle avoidance and following a leader. The obstacle avoidance behavior is simple to define: the robot has to be able to maintain a minimum distance from all other objects in the environment. Although simple in principle, a lot of problems can arise by taking into account the physical constraints of the robot, such as the limited perception (sensors' detection range, precision, accuracy, sampling time, etc.) and limited effector action (speed, acceleration, torque, force-stress, etc.). These factors together with the nature and structure of the environment play a very important role in the design of an effective control strategy not only for obstacle avoidance, but also for any other behavior (simple or complex). The control law of the obstacle avoidance behavior uses the infrared sensor's readings to compute appropriate speeds for the two motors of the Koala robot (figure 3). This data is sufficient to ensure that the robot will not come in contact with any object in the environment. If no obstacle are in sight, the robot just cruises forward at a given constant speed.

The behavior of following a leader robot is more difficult to perform just from infrared sensors' readings (figure 3), because objects from the environment cannot be distinguished from the leader just from this data. On the other hand the control program that uses only the infrared sensors is much more simple and easier to implement.

In this paper, experiments are carried out to show that membrane controllers are viable control strategies for robots operating in a semi-structured office-like environment.

In previous work, a proportional controller for the follower behavior was designed using Numerical P Systems, for Khepera III and e-puck robots [2]. The membrane structure is illustrated in figure 1.

The control law implemented by the membrane structure in figure 1 is the following:

$$\begin{aligned} lw = {} & CruiseSpeedLeft - k_{Dist} * (ref_{Dist} - 0.5 * (s_{Dist1} + s_{Dist2})) \\ & + k_{Heading} * (ref_{Heading} - (s_{R1} + s_{R2} - (s_{L1} + s_{L2}))) \end{aligned} \tag{9}$$

$$\begin{aligned} rw = {} & CruiseSpeedRight - k_{Dist} * (ref_{Dist} - 0.5 * (s_{Dist1} + s_{Dist2})) \\ & - k_{Heading} * (ref_{Heading} - (s_{R1} + s_{R2} - (s_{L1} + s_{L2}))) \end{aligned} \tag{10}$$

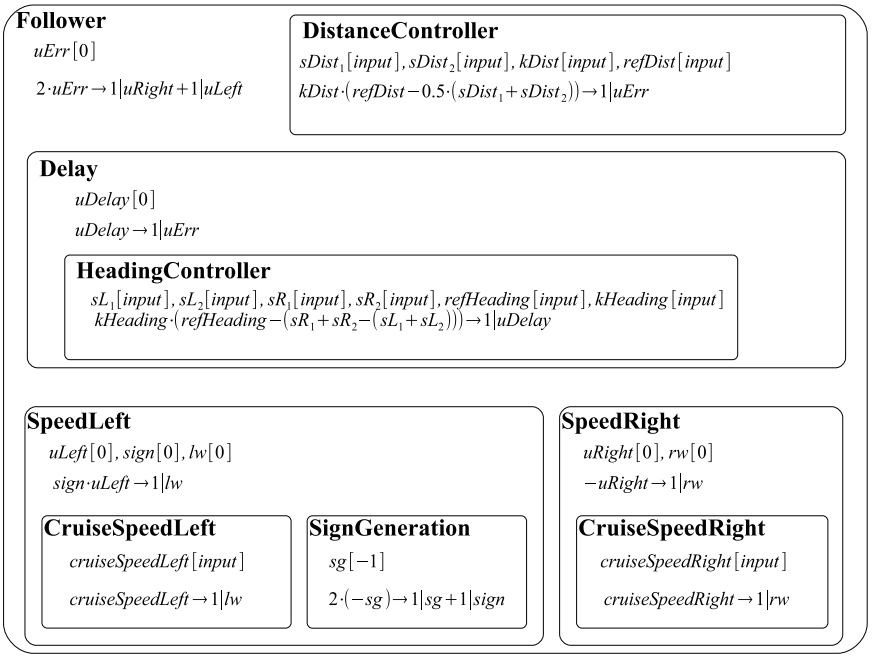

Fig. 1. A proportional controller for the follower behavior, implemented with NPS, for Koala III and e-puck robots

The follower controller was adapted to work with Koala robots, taking into account the sensors' placement on this type of robots. Therefore, we can prove that membrane controllers can be adapted to work on any type of robotic platforms (figure 2).

The control law for the follower behavior of Koala robot is:

$$lw = CruiseSpeedLeft - k_{Dist} * (ref_{Dist} - \frac{1}{6}\sum_{i=1}^{6} s_{Disti}) + k_{Heading} * (ref_{Heading} - (\sum_{i=1}^{6} s_{Ri} - \sum_{i=1}^{6} s_{Li})) \quad (11)$$

$$rw = CruiseSpeedRight - k_{Dist} * (ref_{Dist} - \frac{1}{6}\sum_{i=1}^{6} s_{Disti}) - k_{Heading} * (ref_{Heading} - (\sum_{i=1}^{6} s_{Ri} - \sum_{i=1}^{6} s_{Li})) \quad (12)$$

The NPS structure tested on Koala was computed using SimP simulator proposed in [4]. The follower robot behavior was simulated using Webots simulator and then tested on real Koala robots (figure 3).

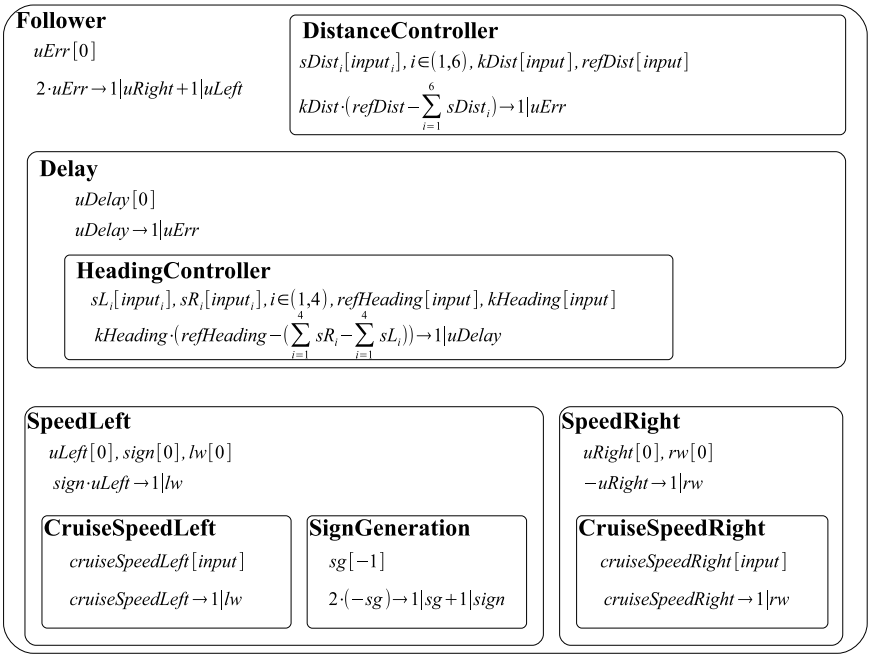

Fig. 2. A proportional controller for the follower behavior, implemented with NPS, for Koala robot

Fig. 3. Koala robot

An equivalent membrane controller using ENPS model has been created as well and is shown in figure 4. The structure of the ENPS controller is simpler and easier to understand and use. It also has less rules and membranes.

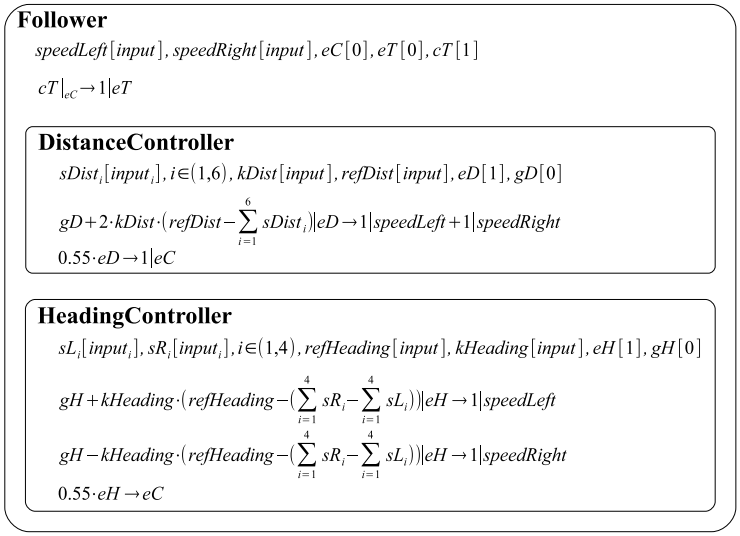

Fig. 4. A proportional controller for the follower behavior, implemented with ENPS, for Koala robot

In the experiments the leader Koala robot performs different predefined motions and also obstacle avoidance behavior. The membrane controller for obstacle avoidance was adapted from the one in [5] and is shown in figure 5. It is based on the ENPS model and the only thing that needed to be changed was to add more membranes for the extra infrared sensors of the Koala robot and the weight of each sensor, which has a different placement than on the other robots. In [5] it is presented how the membrane controller for obstacle avoidance works and what all variables are used for.

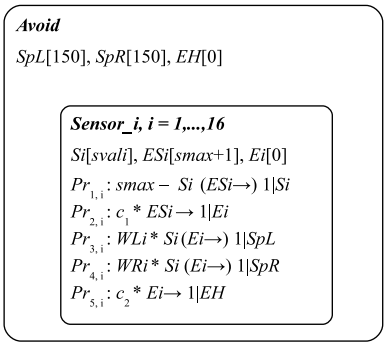

Fig. 5. A controller for the obstacle avoidance behavior, implemented with ENPS, for Koala robot

3 Experiment setup and performance analysis

As stated above, experiments were performed in the WeBots robotic simulator and in a real world setting. In both cases, the environment is a semi-structured office-like environment with or without obstacles. Both obstacle avoidance and follow a leader behaviors were tested, first separately and then together (the leader was performing obstacle avoidance).

All experiments used the SimP membrane simulator to execute the membrane controller that drive the robots. SimP[3] is described in [4] and can be used standalone or as a library. For the experiments on the real Koala robots, a server version was developed in order to respond to the robots' queries. In this setup, a TCP/IP connection is established from each robot to a host machine (in this case a laptop) through a wireless router. The robots run a program that queries the SimP server for the current motor commands based on the sensors' readings. This is done, because SimP is implemented in Java and the robots do not posses the necessary computational capabilities in order to run a Java Virtual Machine.

The performance of the controllers is analyzed based on the speed profiles of the leader and follower robots and also based on the duration of a cycle, the execution time of a membrane controller. The two behaviors were tested separately and together in different scenarios as follows.

Figures 6, 7, 8, 9 show the speed profiles of the leader and follower robots in simulated experiments. These figures are the result of three experiments for the following the leader behavior.

In the first experiment, the leader robot has a forward constant motion with a cruising speed (15) different then that of the follower robot (12). It is clear from figure 6 that the follower matched the speed of the leader and thus maintaining the desired distance to the leader.

In the second experiment, the leader robot has a forward variable motion. The speed of the leader oscillates around the cruising speed of 15 with amplitude $A = 10$ and frequency of about $f = 1.6e - 4$ (equation 13) . Figure 7 shows that the follower is able to match the speed of the leader in this experiment as well. However, a slight phase shift and amplitude difference can be observed in the graph and this is due to the reaction time of the follower, which is at least as long as the duration of a controller cycle. The cruising speed of the follower is also 12 in this experiment.

$$SpeedLeft = CruiseSpeed + A \cdot \sin(2\pi \cdot f \cdot t) \tag{13}$$

$$SpeedRight = CruiseSpeed + A \cdot \sin(2\pi \cdot f \cdot t) \tag{14}$$

In the third experiment, the leader has a variable sine motion. The speed is computed as composition of the constant forward cruising speed and a variable

[3] For the Java binary program (.jar) please contact A.B. Pavel at anabrandusa@gmail.com

turning speed (equation 15). The speed of the left motor is phase shifted by π from the right one. In this case, the follower is still able to follow the leader. However, there is a big difference in the speed profiles of the two robots. These are due to the fact that the Koala robots are square; when the leader turns, its back gets closer to the follower even though the leader is moving away. This is why the follower has to slow down when the leader changes direction and this can be seen in figures 8, 9.

$$SpeedLeft = CruiseSpeed + A \cdot \sin(2\pi \cdot f \cdot t) \tag{15}$$

$$SpeedRight = CruiseSpeed - A \cdot \sin(2\pi \cdot f \cdot t) \tag{16}$$

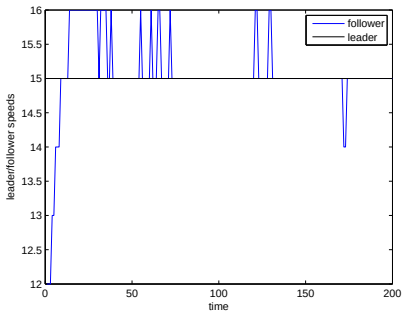

Fig. 6. Speeds of the leader and follower during the forward constant motion

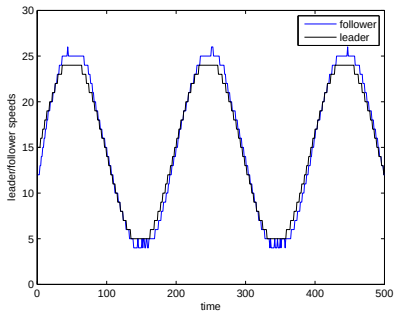

Fig. 7. Speeds of the leader and follower during the forward variable motion

The next figures were obtained from experiments in a real world experiments. Figure 10 shows the speed profile for obstacle avoidance controller. It can be noted that the peaks in the graph correspond to the reaction of the robot when obstacles are detected. The Koala robot was able to avoid all obstacles in the environment.

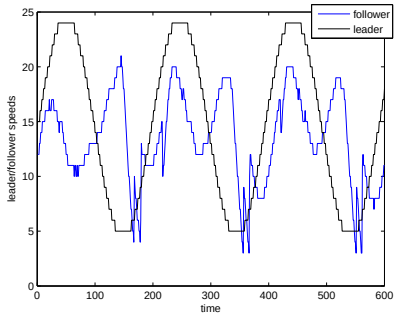

Fig. 8. Speeds of the leader and follower during the sine motion on the left wheels

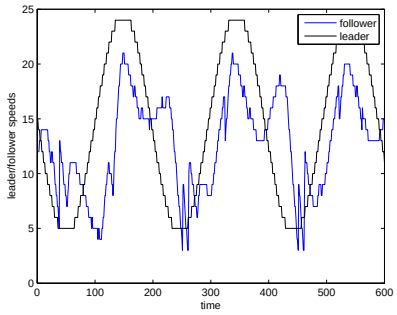

Fig. 9. Speeds of the leader and follower during the sine motion on the right wheels

In figure 11, the speed profiles of a follower robot is shown. The leader has a variable sine motion (equation refeq:leader-sine). It can be seen that the follower robot is able to match the movement of the leader and to follow it.

In the last experiment, the two behavior are used together: the leader performs obstacle avoidance while the other robot follows it. The speed profiles of both robots are presented in figures 12 and 13. The follower is able to keep track of the leader. It is important to note, that the follower must not get to close to other objects in the environment, because it can not distinguish them from the leader robot, only based on the infrared sensors' readings.

In the last figure (figure 14) the execution time of the two membrane controllers in each cycle is shown. It can be seen that the execution time is very stable and small. The first cycles take more time, however, due to initialization of the Java environment. After that, there are no significant peaks in the execution time for the proposed membrane systems.

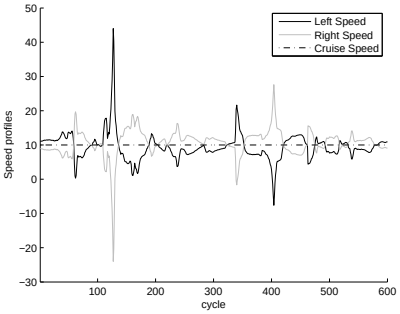

Fig. 10. avoid speed profiles

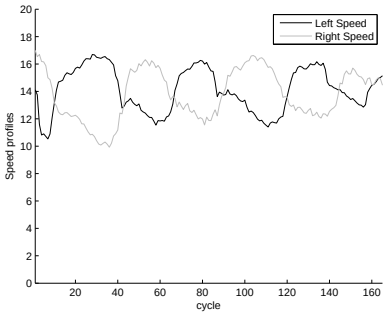

Fig. 11. follow sine speed profiles

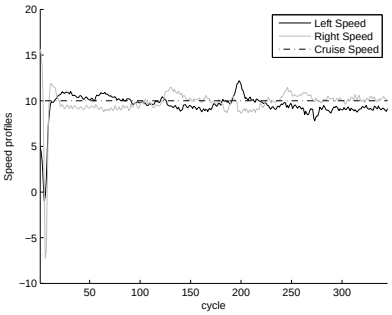

Fig. 12. follow avoid speed profiles

4 Conclusions

In this paper, we have proven that NPS and their extension, ENPS, are a flexible and scalabale modeling tool which can be succesfully used to design robot controllers. Among the advantages of using this computational paradigm in robotics

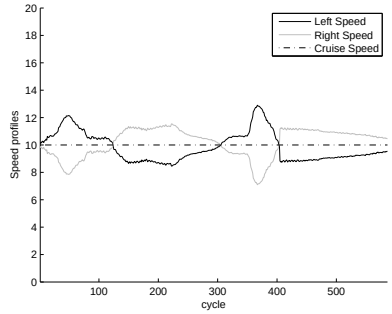

Fig. 13. leader avoid speed profiles

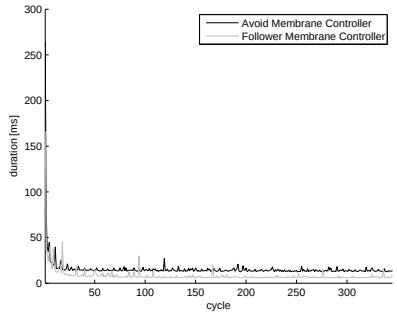

Fig. 14. Avoid/Follow cycle duration

we mention the parallel nature of the model and the possibility of encapsulating behaviors and functionalities as modules which can be executed in parallel. The membranes are independent from the control program of the robotic system and can be easily adapted to other types of robots or applications by only modifying some parameters in the xml files which store the membrane structures.

Future work includes extending the current membrane controllers to other robots, but also to implement other behaviors and functionalities using ENPS model.

References

1. Buiu, C., Pavel, A., Vasile, C., Dumitrache, I.: Perspectives of using membrane computing in the control of mobile robots. In: In Proc. of the Beyond AI - Interdisciplinary Aspect of Artificial Inteligence Conference, Pilsen, Czech Republic. pp. 21–26 (December 2011)
2. Buiu, C., Vasile, C.I., Arsene, O.: Development of membrane controllers for mobile robots. Information Sciences 187, 22–51 (March 2012), doi: 10.1016/j.ins.2011.10.007

3. Pavel, A., Arsene, O., Buiu, C.: Enzymatic numerical P systems - a new class of membrane computing systems. In: The IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2010) Liverpool. pp. 1331–1336 (September 2010)
4. Pavel, A.B.: Membrane controllers for cognitive robots. Master's thesis, Department of Automatic Control and System Engineering, Politehnica University of Bucharest, Romania (February 2011)
5. Pavel, A.B., Buiu, C.: Using enzymatic numerical P systems for modeling mobile robot controllers. Natural Computing (in press), doi: 10.1007/s11047-011-9286-5
6. Pavel, A.B., Vasile, C.I., Dumitrache, I.: Robot localization implemented with enzymatic numerical P systems (submitted)
7. Păun, G., Paun, A.: Membrane Computing and Economics: Numerical P Systems. Fundamenta Informaticae pp. 213–227 (2004)
8. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)