ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2019.1.039

NEURO-EVOLUTION OF MOBILE ROBOT CONTROLLER

Ivan Sekaj[1,✉], **Ladislav Cíferský**[2], **Milan Hvozdík**[2]

[1]Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia
[2]Institute of Informatics and Mathematics s, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia
ivan.sekaj@stuba.sk✉, xciferskyl@stuba.sk, xhvozdik@stuba.sk

Abstract

We present a neuro-evolution design for control of a mobile robot in 2D simulation environment. The mobile robot is moving in unknown environment with obstacles from the start position to the goal position. The trajectory of the robot is controlled by a neural network – based controller which inputs are information from several laser beam sensors. The learning of the neural network controller is based on an evolutionary approach, which is provided by genetic algorithm.

Keywords: *mobile robot, 2D environment, neural network, evolutionary-based learning, genetic algorithm.*

Received: 11 April 2019
Accepted: 5 June 2019
Published: 24 June 2019

1 Introduction

Artificial neural networks (ANN) are used besides many other practical applications as classification, recognition, approximation, etc. also in the control engineering domain as controllers. Under controller we understand generators of control value for controlling continuous-time dynamic systems, robotic systems, logical systems, etc. The advantage of using ANN in comparison to other controller design approaches is the ability to deal with non-linear systems, systems with complex internal structure and complex behaviour. The obvious way in applying ANN in most of application is the use of known input/output data which were obtained during observation of the system behaviour during its normal operation in history (supervised learning, reconstruction of the reality). But obtaining input/output data of a future control process is normally not possible. The (sub)optimal behaviour of the designed controller is a-priori unknown. Known is only the desired closed-loop behaviour, or the desired behaviour of the controlled object respectively.

The aim of this project is to design an ANN-based controller of mobile robot in 2D simulation environment. There exist several approaches for design of mobile robot trajectories, most of them do not use ANNs [1, 2, 3, 4, other]. Our goal is to introduce an evolutionary-based way for designing a direct ANN-based mobile robot controller.

In the second section of this article we explain the mobile robot and its ANN controller architecture with the evolution-based parametrisation. In the third section we present the obtained experimental results.

2 The ANN-based mobile robot controller and its training

2.1 Mobile Robot

The mobile robot considered here is in a conventional 4-wheel configuration with turnable front wheels as depicted in Fig.1. The information about the surrounding environment front of the robot is provided by sensors, which measure the distances between obstacles and the front of the robot as well as the type of observed obstacles (wall, goal, etc.) in several beams (Fig. 1a). The control value represents the steer angle φ which is the deviation between the robot back-front axes and wheel orientation axes (Fig. 1b).

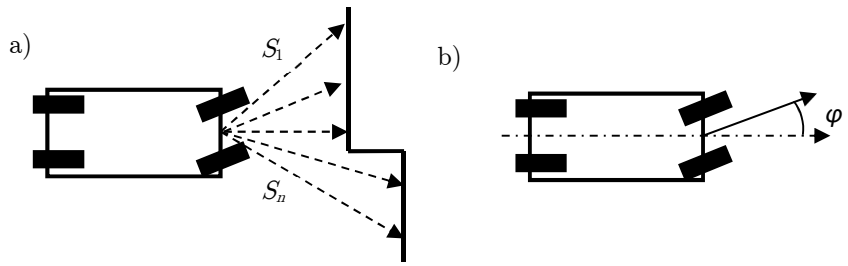

Figure 1: Mobile robot with 4 wheels, a) sensor beams measuring distances between obstacles, b) steer angle φ.

In each control sampling period, the robot evaluates the distances from obstacles located in front of the robot, calculates the next value of the steer angle φ and moves a single step in this direction. Without loss of generality, let us consider here, that the velocity of the robot is constant and do not change during the motion.

2.2 The ANN-based controller architecture

The control algorithm contains following three steps in each sampling period: 1. measuring of obstacle distances and identification of their type (sensors $S_1 - S_n$), 2. calculation of the new steer angle φ, 3. single step move in the direction φ. Calculation of the steering angle in each sampling period is provided by the ANN. For this purpose, the multilayer perceptron network (MLP) has been considered. We have used the feed-forward MLP network as well as a recurrent MLP network. Block scheme of the used recurrent MLP network is depicted in Fig. 2. The input values of the ANN ($X_1 - X_n$) are real values of distances and obstacle types obtained from sensors $S_1 - S_n$. Two hidden layers of neurons have been used with weight matrixes W_{H1} including weights $w_{H1,1,1}, \dots, w_{H1,N2,N1}$ and W_{H2} with weights $w_{H2,1,1}, \dots, w_{H2,O,N2}$. The neurons of hidden layers in Fig. 2 are represented by dark circles except the right most one which is the output neuron. The output layer weight vector W_o include weights $w_{O,1,1}, \dots, w_{O,1,N1}$. Recurrent connections contain single time-step delays D (z^{-1} in the z-transform). Block scheme of a neuron (each dark circle in Fig. 2) is shown in Fig. 3. The hyperbolic tangent is used in case of the activation function *f(a)*.

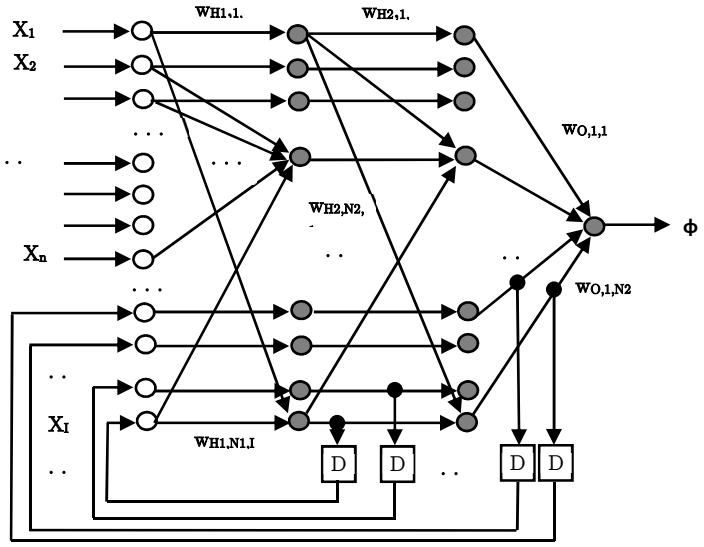

Figure 2: Recurrent MLP network with 2 hidden layers and recurrent connections containing single time-step delays D.

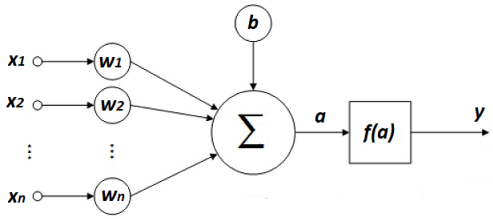

Figure 3: Block scheme of a single neuron of the MLP network, *f(a)=tanh(a)*.

2.1 The ANN-based controller architecture

The unknown parameters of the ANN controller which are objects of the optimisation process are the synaptic weights $w_{Hi,j,k}$, where i is the order number of the layer (1, 2 are hidden layers, 3 is output layer), k is the number of the neuron in the layer and j is number of the connected neuron in the next layer. For the search of their optimal values an evolutionary approach based on the genetic algorithm (GA) has been used. Each individual of the GA population (a chromosome) is a vector of all weights of the ANN in form

$$ch = \{W_{H1}, W_{H2}, W_O\} = \{w_{H1,1,1}, \dots, w_{H1,N2,N1}, w_{H2,1,1}, \dots, w_{H2,O,N2}, w_{O,1,1}, \dots, w_{O,1,N2}\} \quad (1)$$

The goal of the task is to find such optimal values of the chromosome ch_{opt} which minimise the fitness function $C(ch)$:

$$ch_{opt} = \arg\min C(ch) \quad (2)$$

The fitness function represents the measure of performance of mobile robot behaviour considering distance to the target, trajectory length, number of collisions with obstacles or other metrics respectively. The used genetic algorithm consists of following steps:

1. random initialization of N chromosomes of population P
2. **for** g=1 to G **do** // G = number of generations
   - 2.1. **for** i=1 to N **do**
     - 2.1.1. transform chromosome ch_i to robot model // genotype to phenotype
     - 2.1.2. **simulate** the i-th robot trajectory
     - 2.1.3. **evaluate** the fitness function F(ch_i) of the i-th robot // according (3)

   end for // 2.1

   2.2. P1=**select**$_{\mathbf{SUS}}$(P) // Stochastic Universal Sampling (SUS) [5, 6], 40% of population

   2.3. P2=**select**$_{\mathbf{SUS}}$(P) // Stochastic Universal Sampling (SUS), 40% of population

   2.4. P3=**select**$_{\mathbf{rank}}$(P) // Ranking selection, 2% of population

   2.5. P4=**select**$_{\mathbf{rnd}}$(P) // Random selection, 18% of population

   2.6. P1=**crossover**(P1) // single point crossover

   2.7. P2=**mutate**(P2) // mutation rate = 0.1

   2.8. P=**unify**(P1,P2,P3,P4) // new population

   end for // 2
3. solution=best individual of the P

3 Experimental results

In the experimental setup of the robot five sensor beams were considered with angles {-30°, -15°, 0°, +15°, +30°}. All their corresponding ANN inputs are normalized in range [0,1]. The ANN output (steer angle φ) is in range $-\pi/2 < \varphi < \pi/2$. Next two inputs are used for identification of the type of obstacle: wall, target. The ANN architecture contains 7 neurons in the input layer, two hidden layers each with 12 neurons and a single neuron in the output layer. If variable velocity of the robot is required, an additional neuron in the output layer has to be added with the output value – velocity. The fitness function is considered in the following form

$$F = c_1 DTT + c_2 LT + c_3 C + B \tag{3}$$

DTT is the euclidean distance between the robot and the target, LT is the length of the robot trajectory, C is number of collisions between the robot and the obstacles and B is bonus when robot is reaching the target. Weight constants were set to $c_1 = 100, c_2 = 100, c_3 = 150, B = 5000$. The population size used was set between 80 and 160 individuals depending on the problem complexity.

Several experiment scenarios were considered. The simplest case was learning of the robot to move from start position to goal position in the environment without obstacles (Fig. 3). To ensure good generalisation ability of the robot during the learning process five different start/goal position configurations were considered for a single fitness function calculation of each individual of the population. After evolution the robot was able to move from arbitrary starting point to arbitrary goal point. Note that when only a single start/goal configuration is considered in the fitness evaluation the robot will learn only this specific trajectory and it would not be able to deal with a-priori unknown situations during normal operation. A more complex example is shown in Fig. 4 where the environment contains obstacles. In Fig. 5 the fitness function graph is depicted for two independent GA runs. In Fig. 6 graphs of the four used partial criterions of the fitness function (3) are separately depicted: 1. number of collisions with obstacles, 2. distance to target, 3. bonus when reaching target and 4. length of trajectory. Sixty generation of evolution were needed to learn the robot to move in the environment with obstacles. The most complex scenario is shown in Fig. 7. The success rate of reaching the target after 200 generation of evolution was 75 %. Example of an unsuccessful training run is shown in Fig.8. An increase of the success rate can be obtained using more neurons in the hidden layers, the use of recurrent links in the ANN (see Fig. 2) as well as by applying more generations of the GA run. The experiments have been developed in the Matlab programming environment [7].

4 Conclusion

A mobile robot movement control strategy based on the neuro-evolution approach has been proposed. The genetic algorithm is able to design parameters of an a-priori defined architecture of the neural network controller. Feed-forward as well as recurrent ANNs have been parametrised using the GA. The obtained

experiments have shown that ANNs with two hidden layers are able to control mobile robots in non-trivial environments. The experiments have shown that neuro-evolution is an efficient approach for solving unsupervised learning problems in mobile robot navigation applications.

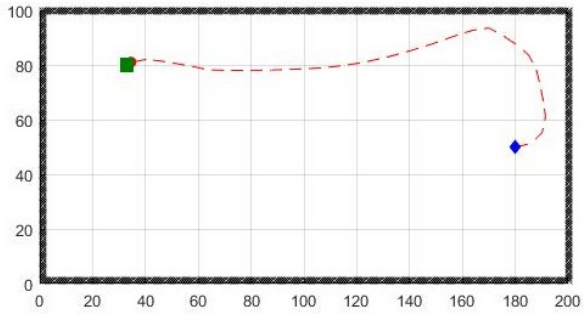

Figure 3: Robot trajectory from start (blue diamond) to goal (green square).

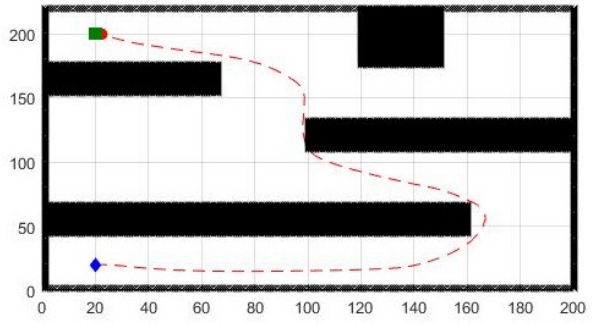

Figure 4: Robot trajectory in environment with obstacles

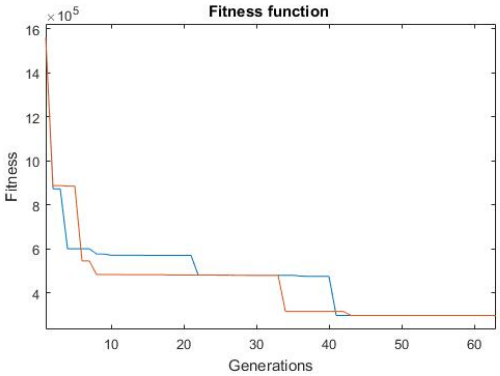

Figure 5: Graphs of fitness function for two independent GA runs.

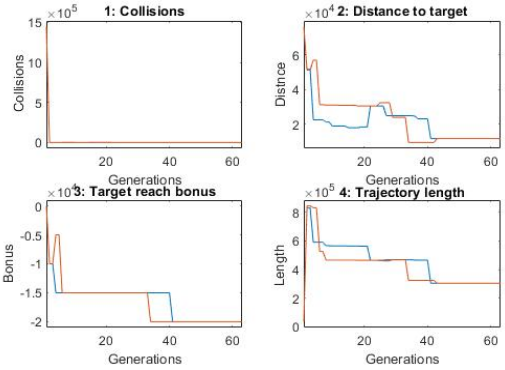

Figure 6: Graphs of partial criterions of the fitness function for two independent GA runs

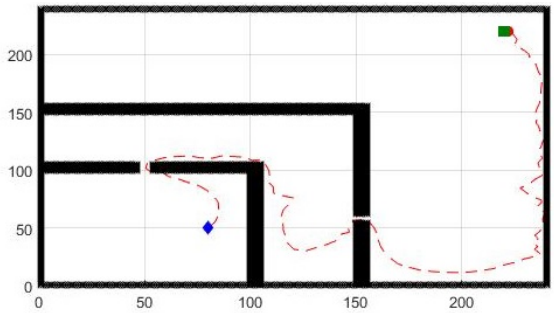

Figure 7: Robot trajectory in a non-trivial scenario.

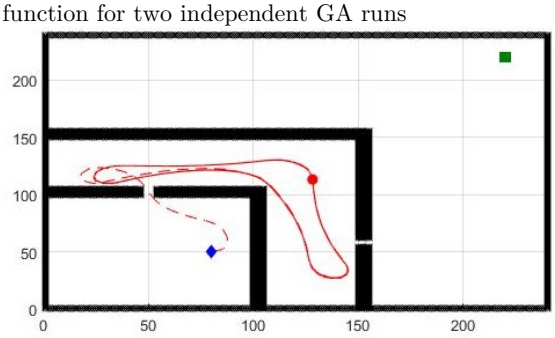

Figure 8: Unsuccessful evolution result. The robot (red dot) starts to move in periodic loops

Acknowledgement: This project was supported by the grant of the Slovak research and grant agency, grant No. VEGA 1/0475/16.

References

[1] Siciliano, B. and Khatib, O. 2008. *Springer handbook of robotics*, Springer-Verlag Berlin Heidelberg.

[2] Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., and Jurišica, L. 2014. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96, pp. 59–69.

[3] Babinec, A., Duchoň, F., Dekan, M., Pásztó, P., and Kelemen, M. VFH* TDT (VFH* with Time Dependent Tree): A new laser rangefinder based obstacle avoidance method designed for environment with non-static obstacles. *Robotics and autonomous systems*, 62, 8, pp. 1098–1115.

[4] Duchoň, F., Huňady, D., Dekan, M., and Babinec, A. 2013. Optimal navigation for mobile robot in known environment. *Applied Mechanics and Materials* 282, pp. 33–38.

[5] Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolutionary Programs.* Springer-Verlag Berlin Heidelberg.

[6] Eiben, A. E. and Smith, J. E. 2003. *Introduction to Evolutionary Computing.* Springer-Verlag Berlin Heidelberg.

[7] Matlab R2017b. 2017. The MathWorks, Inc. www.mathworks.com