This article can be cited as J. Vascak and M. Pal'a, Adaptation of Fuzzy Cognitive Maps for Navigation Purposes by Migration Algorithms, International Journal of Artificial Intelligence, vol. 8, no. S12, pp. 20-37, 2012.

Adaptation of Fuzzy Cognitive Maps for Navigation Purposes by Migration Algorithms

Ján Vaščák and Martin Paľa

Department of Cybernetics and Artificial Intelligence
Technical University of Košice
Letná 9, 042 00 Košice, Slovakia
jan.vascak@tuke.sk, martin.pala@tuke.sk

ABSTRACT

Fuzzy Cognitive Maps (FCM) represent not only a user-friendly knowledge representation but also a convenient means for simulation of dynamic systems and decision-making support. Concerning the nature of robotic systems FCM seem to be convenient in using mainly on upper decision levels. However, FCM strike on problems of their design. Beside manual approach, which is limited by the number of nodes and their connections, various adaptation methods have been proposed. This paper gives a short summary of these methods dividing them into Hebbian-based and evolutionary-based approaches. Further, it presents a new adaptation of the so-called Self-Organizing Migration Algorithms (SOMA) for purposes of FCM design, which is compared also to other methods like particle swarm optimization, simulated annealing, active and nonlinear Hebbian learning on experiments with catching targets for future purposes of robotic soccer. Obtained results are compared where advantages of the proposed method are apparent and in the conclusions their properties are summarized. Besides, a new modification of FCM with active inputs is presented that is able to receive data from sensors in each time step.

Keywords: fuzzy cognitive maps, self-learning, migration algorithm, navigation.

2010 Mathematics Subject Classification: 68T05, 68T40, 93C85.

ACM Computing Classification System: I.2.4, I.2.6, I.2.9, I.6.5.

1 Introduction

Automatic navigation of vehicles or robots represents a special kind of decision processes, which can be basically divided into *motion planning* and *reactive navigation* (LaValle, 2006). The first category is obviously connected also with strategy choice, e.g. in robotic soccer (Vaščák and Hirota, 2011), where a convenient trajectory is constructed by defined limitations and requirements. The second category is in the sense of its original purpose, whose task is to track the prescribed trajectory and eventually to react to unpredictable situations like for instance obstacles. In both cases decision making is performed and therefore the first objective is to find proper means for description of decision processes as well as solving given tasks.

If we take into account that decision processes are typical with complex decision (implication) chains, which also create closed loops then using conventional rule sets will be unsatisfactory or at least their structure will be incomprehensible for any human. Therefore *Fuzzy Cognitive Maps* (FCM) offer a very suitable means for clear knowledge description as well as its efficient processing with the help of matrix operations. Experience from numerous applications has shown they are able thanks their nonlinearity to capture more information than another mapping methods. They are dynamic, flexible and able to extract hidden knowledge (Bertolini and Bevilacqua, 2010). From this reason already in the 90-ties FCM found their use in administrative sciences, game theory, information analysis, cooperation man–machine, distributed group decision and technology management (Aguilar, 2005). Continuously they have penetrated also into the area of control and robotics in the last decade, e.g. (Golmohammadi, Azadeh and Gharehgozli, 2006; Vaščák and Madarász, 2010).

However, FCM have some drawbacks, too. Firstly, they tend to undesirable states. Secondly, they need to be constructed by some experts from given area, which is a typical problem of all conventional fuzzy systems: they are not able to self-learn. This is the main motivation for development of learning mechanisms for FCM, especially in the last ten years, which can be divided into two basic groups — *Hebbian-Based Learning* (HBL) and *Evolutionary-Based Learning* (EBL). Obviously, as the design of adaptation approaches is much more difficult than in the case of conventional rule bases due to complex structure and variability of FCM (Johanyák and Kovács, 2006) at least the definition of nodes is done manually by experts and adaptation is limited to adjusting relations, i.e. graph edges. As the structure of FCM resembles to neural networks the first (historically older) group of HBL comes just from this area. FCM is an oriented graph and the basic task of learning is to determine relations among nodes and subsequently their strengths in such a way that the resulting FCM would reach rather a stable state than a model describing a given system. For such purposes HBL and its numerous modifications seem to be the most proper candidates for learning, e.g. *active* and *nonlinear Hebbian learning* (HL) (Papageorgiou, Stylios and Groumpos, 2006) or its improved version (Li and Shen, 2004). The advantage of this approach is that mostly they do not need any historical training data. Learning can be done using only current data and in many cases because of relatively simple learning algorithms it can be performed on-line. However, HBL exhibits also drawbacks. The accuracy of obtained results is often unsatisfactory. Mostly we need to obtain not only a stable state but also desired values, i.e. it is not a task of unsupervised but supervised learning, which is not the case of HBL. In (Vaščák and Madarász, 2010) it is shown how under some limiting conditions also supervised learning can be utilized for FCM as well as a comparison with other HBL methods.

On the contrary, EBL methods use historical data with the aim to find a model describing a given system. Based on the axiom of evolutionary optimization many times repeated computation should lead to results near the optimum and thus it should reach a stable state, too. However, the tax is high computational complexity and hereby any on-line is excluded. In last ten years especially this group of learning algorithms has been researched and various approaches have been tested starting from conventional genetic algorithms (Koulouriotis, Diakoulakis and Emi-

ris, 2001), through real-coded genetic algorithms (Stach, Kurgan, Pedrycz and Reformat, 2005) using real values of genes instead of binary coding to such parts of evolutionary computation like *Particle Swarm Optimization* (PSO) (Papageorgiou, Parsopoulos, Stylios, Groumpos and Vrahatis, 2005). In a broader sense of EBL as a part of metaheuristic optimization for our purposes we could count into this group *Simulated Annealing* (SA) (Ghazanfari, Alizadeh, Fathian and Koulouriotis, 2007), too. Just PSO and SA exhibit very promising experimental results and will be further subjects of comparison.

Another optimization approach, which can be classified to swarm intelligence, is *Self-Organizing Migration Algorithm* (SOMA), based on intelligent cooperation as for instance wolf packs, whose basic ideas were firstly formulated in (Zelinka, 2002). It has been used in various areas like diagnostics, tuning or scheduling (Vaščák, 2005). In (Nolle, Zelinka, Hopgooda and Goodyear, 2005) several means like differential evolution, SA and SOMA were compared on a task of sensor tuning. SOMA brought the best results. Based on repeatedly excellent experience with using SOMA the idea arose to adapt this approach for constructing FCM being able to navigate vehicles. This supposition has been strengthened by a fact that scheduling is incorporated into navigation, too.

After a short introduction related to advantages of FCM in navigation tasks and their adaptation possibilities the section 2 describes basic notions, definition and processing of FCM. The following section 3 describes the basic ideas and function of SOMA in detail because it is quite newel in this area. The previous sections are necessary for description of a new FCM modification as well as application of SOMA for navigation, which is contented in the section 4. In the section 5 two experimental scenarios are realized and obtained results from several adaptation methods are compared and evaluated. Finally, some concluding remarks close this topic in the last section.

2 Fuzzy Cognitive Maps

The notion *Cognitive Map* (CM) was introduced by a political scientist Robert Axelrod primarily to model social processes (Axelrod, 1976). It is an oriented graph where its nodes represent notions and edges are causal relations. Mostly, notions are states or conditions and edges are actions or transfer functions, which transform a state in a node to another one in another node. CM is able to describe complex dynamic systems. It is possible to investigate cycles, collisions, etc. and to define strengths (weights) of relations, too. Originally, they were represented by three values $-1, 0$ and 1. Another advantage is its human-friendly knowledge representation and ability to describe various types of relations (in a more detail see $\mathrm{e.g.}$ (Groumpos, 2010)).

FCM represents an extension of CM and was proposed by Kosko in 1986 (Kosko, 1986). The extension is based on strength values that are from an interval $[-1; 1]$ as well as the nodes can be represented by activation values from an interval $[0; 1]$ or by membership functions as well. Strengths after their combining correspond to rule weights in rule based systems, too.

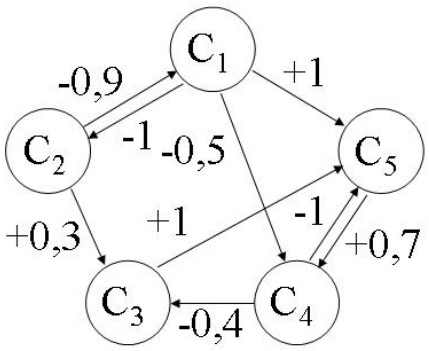


Figure 1: An example of FCM.

There are two basic formal definitions of FCM (Chen, 1995) and (Stach et al., 2005). Further, the definition by Chen will be used where FCM is defined as a 4-tuple:

$$FCM = (C, E, \alpha, \beta), \tag{2.1}$$

where:

- C - finite set of cognitive units (nodes) described by their states $C = \{C_1, C_2, \ldots, C_n\}$;
- E - finite set of oriented connections (edges) between nodes $E = \{e_{11}, e_{12}, \ldots, e_{nn}\}$;
- α - mapping $\alpha : C \to [0; 1]$ (originally proposed as $[-1; 1]$);
- β - mapping $\beta : E \to [-1; 1]$.

In other words, α is a computational method for evaluating numerical values of the set of nodes C. On behalf of correctness, it is necessary to mention that a cognitive unit is represented by two values: (a) its symbolic (linguistic) meaning denoted as C_i and (b) its numerical activation value $A_i \in [0; 1]$. C_i represents the qualitative aspect and A_i the quantitative one of a node. However, from simplicity reasons we can omit this distinction and we will use the symbol C_i although further we will handle only with activation values. On the other hand, β represents the way how the weights of edges $e_{ij} \in [-1; 1]$ are determined. The signs in this case define the character of relationship between nodes — either strengthening $(+)$ or weakening $(-)$.

The set of connections E forms the so-called *connection (adjacency) matrix*, which can be used for computing of new activation values of nodes C. For the example in the fig. 1 it will look as:

$$E = \begin{bmatrix} 0 & -1 & 0 & -0,5 & 1 \\ -0,9 & 0 & 0,3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -0,4 & 0 & -1 \\ 0 & 0 & 0 & 0,7 & 0 \end{bmatrix}. \tag{2.2}$$

Cognitive units are in each time step k in a certain state. Using E we can compute their states for next time step $k+1$ and thus repeatedly for further steps. Similarly as for differential equations we can draw phase portraits. To preserve values in prescribed limits a *limiting* (*threshold* or

transform) *function* L is used, too. So we can compute the states C_i for $k+1$ as follows (Stach et al., 2005):

$$C_i(k+1) = L\left(\sum_{j=1}^{n} e_{ij}.C_j(k)\right). \tag{2.3}$$

The formula (2.3) can be generalized for computation of the whole state vector C using a matrix product, i.e. $C(k+1) = L(E.C(k))$. There exists still an incremental modification of (2.3) where $C(k)$ is incremented by computed values, e.g. (Papageorgiou et al., 2005):

$$C_i(k+1) = L\left(C_i(k) + \sum_{j=1}^{n} e_{ij}.C_j(k)\right). \tag{2.4}$$

The primary role of the function L is to keep activation values in the interval $[0; 1]$. A number of functions fulfill this condition. However, as the most advantageous the sigmoid function seems to be (Bueno and Salmeron, 2009). For a special kind of *FCM with active inputs*, which will be described in the section 4, any kind of typical membership functions can be used as L.

3 Self-Organizing Migration Algorithm

SOMA is based on cooperative searching (migrating) the area of all possible solutions, i.e. search area. Individuals are mutually influenced during the search process, which leads to forming or canceling groups of individuals. Such groups organize themselves the movement of individuals. From this reason the adjective 'self-organizing' is contained in its title. SOMA parameters can be roughly divided into two groups:

- *managing parameters* — they influence the quality of search;
- *finishing parameters* — they determine the stopping moment of the algorithm.

Their definition is following:

$Mass$ — coefficient for the *migration vector* $\vec{m}$ (3.3), which defines final position of an individual after one *migration cycle*. If e.g. $Mass = 1$ the individual will stop directly on the leader's position. If $Mass = 2$ the individual will stop on the position $2.\vec{m}$, i.e. the leader will be in the middle between the initial and final position of the individual. It is recommended the $Mass$ value to adjust > 1 to cover the search area by individuals on a larger surface to prevent their skidding into a local extreme;

$Step$ — size of a migration step or mapping. The total number of migration steps is $nms = Mass.\|\vec{m}\|/Step$ where $\|\vec{m}\|$ is the magnitude of $\vec{m}$. The smaller $Step$ the greater the chance to find a significant extreme but also higher computational complexity and vice versa;

PRT — perturbation, a parameter, which modifies the *movement vector* $\vec{m}$ of an individual to the leader;

D — number of optimised variables or arguments of the fitness function. This parameter is directly depended on the solved problem and defined fitness function;

NI — number of individuals (population size). This value depends usually on D and it directly influences the search quality. The greater NI is then the higher possibility will be to find a significant (maybe global) extreme;

M — number of *migration cycles*, which is analogous to the number of populations in genetic algorithms;

AE — the maximum allowed difference between the best and the worst individual in the population. To find really a significant extreme and to prevent divergence from the optimal solution it is necessary to achieve good solution also for other individuals not only for the best one. It means if the real error is smaller than the accepted error then the algorithm will be stopped.

The parameters from $Mass$ to NI belong to the first group and the last two parameters are finishing ones.

One important advantage of this algorithm is based on its ability to process diverse data types of parameters like integers, real or discrete values. They can be mixed mutually, too. SOMA parameters define the structure and universes of discourse for individuals. To generate an initial population the so-called *specimen* S is defined at first:

$$S = \left(s_1^{Type}(s_1^{LL}, s_1^{UL}), \ldots, s_D^{Type}(s_D^{LL}, s_D^{UL}) \right), \tag{3.1}$$

where $Type$ denotes the data type of a parameter, LL and UL are the low and upper limits of the universe of discourse, respectively. These intervals of values represent permitted parameter values or from other point of view physical limitations of a given application. The population (real individuals) will be generated by:

$$P^0 = \left\{ x_{1,1}^0, \ldots, x_{i,j}^0, \ldots, x_{NI,D}^0 \right\} = \left\{ \mathrm{rnd}(x_{i,j}^{UL} - x_{i,j}^{LL}) + x_{i,j}^{LL} \right\}, \tag{3.2}$$

where P^0 is the initial population and $x_{i,j}$ represents j-th dimension of the i-th individual ($i = 1, \ldots, NI$ and $j = 1, \ldots, D$).

In addition, SOMA uses also operators of perturbation and migration. The perturbation is analogous to the mutation process in genetic algorithms. However, the result of such an operation is not a property change of an individual but its *movement vector* $\vec{m}$ to the leader is perturbed (interfered), i.e. it is not directed to the leader (as seen in fig. 2). The *movement vector* $\vec{m}$ represents the distance between starting point I_0 of a given individual I and the leader L, i.e. in the vector description:

$$\vec{m} = \vec{r}_L - \vec{r}_0, \tag{3.3}$$

where $\vec{r}_L$ and $\vec{r}_0$ are vectors of the leader and the starting point of a given individual, respectively.

Hence, the perturbation has following influence on the real position of such an individual in next step $\vec{r}$:

$$\vec{r} = \vec{r}_0 + p.\hat{m} * \vec{v}_{PRT}, \tag{3.4}$$

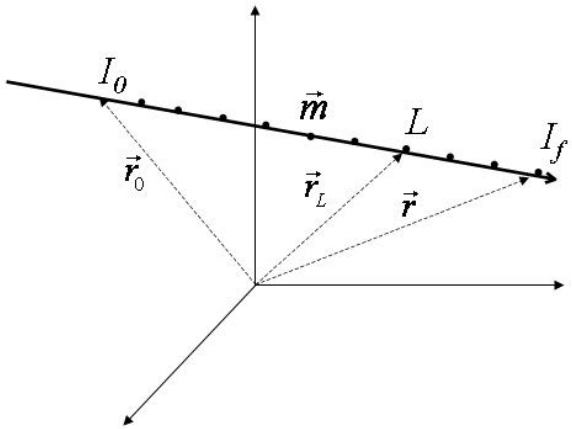


Figure 2: Relations between vectors $\vec{r}_0$, $\vec{r}_L$ and $\vec{r}$ in a 3-dimensional space ($D = 3$); I_0, I_f – starting and final positions of an individual I, L – leader.

where $\hat{m}$ is the unit vector of $\vec{m}$ and p relates to the order of steps k on a path of a given individual I from the starting point I_0 ($\vec{r}_0$) to the final one I_f ($\vec{r}$), i.e. $p = k.Step$, $k = 0, 1, \ldots, nms$ (individual steps in the fig. 2 depicted as bullets •). The elements of the *perturbation vector* $\vec{v}_{PRT}$ are created in each *migration cycle* by a condition: $\text{if } \text{rnd}_j < PRT \text{ then } \vec{v}_{PRT,j} = 1, \text{ else } j = 0$, where rnd_j is a randomly generated number and j is the index for a given property ($j = 1, \ldots, D$). The vector $\vec{v}_{PRT}$ is in reality a mask and the operation $*$ performs pairwise products among individual elements of $\hat{m}$ and $\vec{v}_{PRT}$. If PRT has a small value then $\vec{v}_{PRT}$ will have mostly zeros and the perturbation will affect direct movement of a given individual to the leader, i.e. the *movement vector* $\vec{m}$ will be modified. Only the dimensions where values of $\vec{v}_{PRT,j}$ are adjusted to 1 will not be perturbed and the movement will be similar to the original form of $\vec{m}$ (see fig. 3).

Similarly, migration is analogous to crossover in genetic algorithms. During one *migration cycle* (3.4) is processed in steps, which corresponds to mapping the state space. Although there does not exist any generating new populations but this representation is equivalent to a sequence of descendants depicted in the fig. 3 as bullets (one step – one descendant or one element of given population). Also the best solution will be chosen and after the *migration cycle* the individual will come back to the best position, which corresponds to the selection in genetic algorithms. Generating new populations is substituted by migrating individuals in the state space.

Processing in SOMA depends on the strategy used. There are several possible strategies but the strategy All-To-One is the primary one and the following process will be explained using this kind of strategy. First, after some initializing steps like defining *managing* and *finishing parameters* as well as creating the initial population from the *specimen* each individual will be evaluated by a fitness function. The best individual will be chosen for a leader in next *migration cycle*. After that individuals start to move to the leader in jumps calculated by (3.4) then they will pass it and finish at distances given by parameters $Step$ and $Mass$ (see fig. 4). After each jump

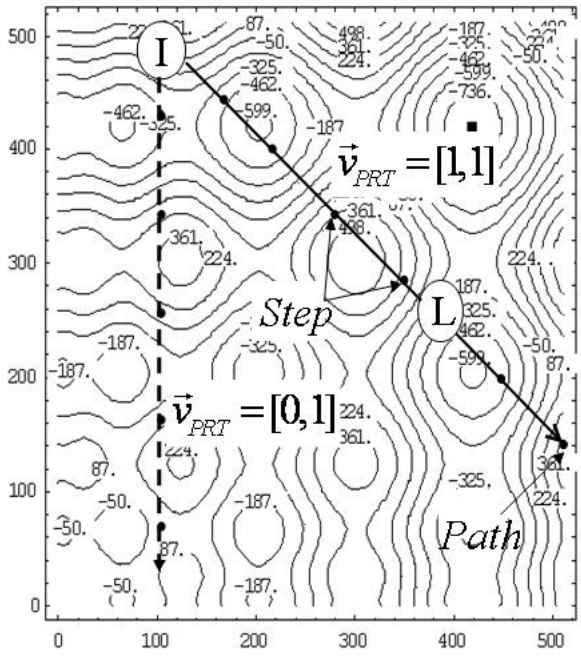


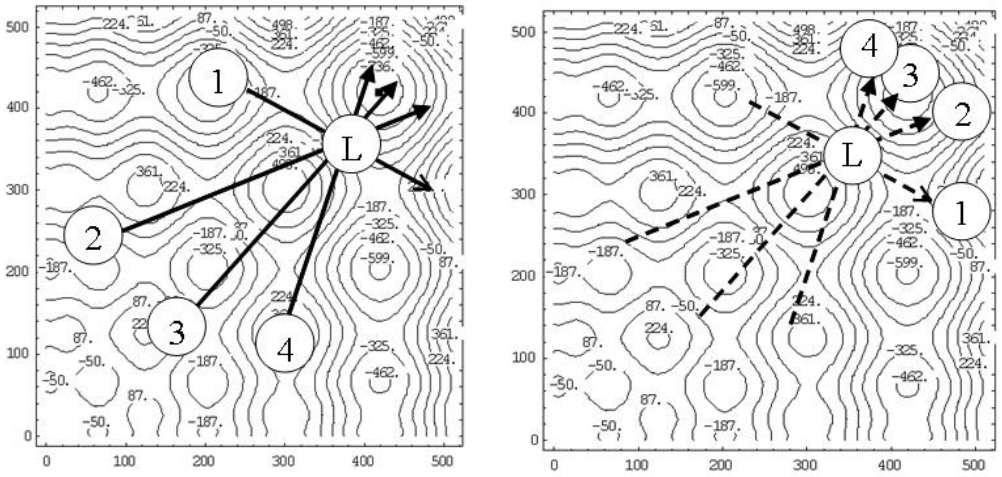
Figure 3: Influence of the *perturbation vector* $\vec{v}_{PRT}$ on the *movement vector* $\vec{m}$ in a 2-dimensional space; I – individual, L – leader, dashed arrow – perturbed, solid arrow – not perturbed.

(step) the individuals will be evaluated by the fitness function. If the evaluation is better than the previous one it will be remembered. After the individuals reach the last jump they will return to new positions where they found the best fitness, i.e. beside the leader individuals will be moved to other positions (fig. 4). In other words, they did searching in the state space and migrated. In such a manner one *migration cycle* has been finished and new one will immediately start after a new leader (the best individual) is determined.

As seen from above the size of population remains the same, even the individuals are the same. No new population will be generated and no selection in the sense of genetic algorithms will be done. The acquired knowledge remains in each individual and we can observe certain learning process during *migration cycles*. The only kind of selection can be observed in choosing the leader, which depends on the quality of an individual (fitness value). The strategy of information interchange in SOMA (similarly also PSO), i.e. cooperation and competition in searching a leader can be withdrawn, in spite of genetic algorithms. However, in contrast to PSO the exploration of leader's surrounding is more systematic and spreading individuals less stochastic. Therefore, a kind of intentional (deterministic) learning is in a greater measure than in conventional genetic algorithms or PSO.

4 Problem Description and Implementation of FCM

The original aim for research was its application in robotic soccer where properties of operating area are known and thus it is quite easy to process colour image segmentation. Such an area



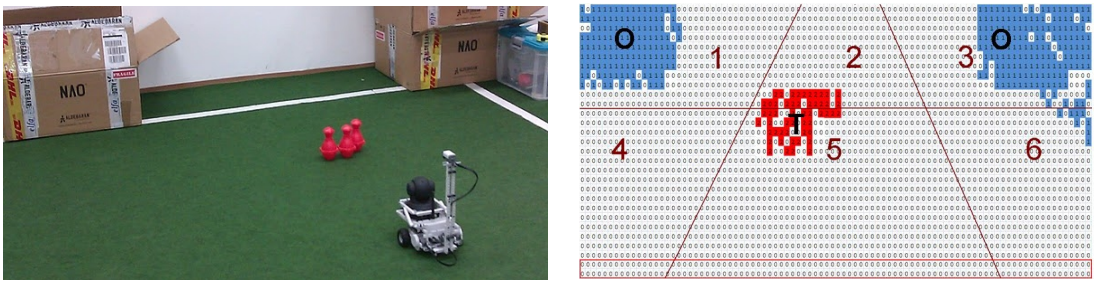
(a) (b)

Figure 4: Migration process of individuals 1 - 4 to the leader L from the positions in (a) to (b) during one *migration cycle*.

consists of several objects characteristic by their unique colours. The ball is orange or red, playground is green and lines are white. All other colours represent obstacles. Further, we will focus our attention on the ball as the main target, which should be chased and catched. In other words, red or similar colour is the target, white and green are free areas and remainder is an obstacle. Thus we get three classes — target, free area and obstacle(s).

As the image is processed as a set of colour pixels we can transform them into elements of a *visual matrix* $M \times N$ where M is the number of pixels along the hight and N number along the breadth of the sensed picture by a camera, which is mounted on the top of our vehicle (robot). The elements will be assigned values by the colours of pixels, i.e. target as 2, obstacles as 1 and free area as 0. As the camera senses images in vertical position it is apparent that higher positioned pixels correspond to more distant objects (*far*) than lower ones (*near*). From this reason several lowest rows (tagged by a red-lined rectangle) represent the area of critical nearness. If there are some obstacles or the target then the vehicle will stop immediately — either because of obstacles or the target has been reached. Further, the matrix is divided into two vertical (*far, near*) and three horizontal (*left, straight, right*) sectors, i.e. in total six (1 – 6). Thus a real image can be transformed into a *visual matrix* as seen on the fig. 5 with a set of skittles as a target. This depiction is for illustration only because fig. 5 a is not taken from the camera on the vehicle, which is visible on the picture and therefore it is slightly different from that one transformed into the *visual matrix*.

Target and obstacles create clusters, which can be described in a simpler way than using a huge *visual matrix*. It can be represented just by sectors 1 – 6 (clusters) in satisfactory quality of accuracy. They could be also linguistically expressed like *far left* for the sector 1 and *near right* for the sector 6. For each sector i the *presence measure* of obstacles pmo_i as well as target pmt_i is defined. It is a weighted average where weights w_{ij} are indirectly proportional to distances d_{ij} between the vehicle and a given pixel, i.e. $w_{ij} = 1/d_{ij}$. If a pixel is occupied by



(a) (b)

Figure 5: Transform of a real image (a) sensed by the vehicle's camera to a *visual matrix* (b) with sectors 1 – 6; obstacles – O, target – T.

the target or obstacle then its *presence value* p_{ij} will be set to 1 else to 0. Thus for n_i pixels of individual sectors ($\sum n_i = M \times N$) we get in total 12 pm values, which represent a part of the state vector C for proposed navigation FCM:

$$pmo_i/t_i = \frac{\sum_{j=1}^{n_i} w_{ij}.p_{ij}}{\sum_{j=1}^{n_i} w_{ij}}. \tag{4.1}$$

(4.1) secures the pm values will stay in the interval $[0; 1]$ as well as they will be influenced by the distance between individual areas and the vehicle. Thus more distant areas will be less important than nearer areas. Near objects force the vehicle to undertake more drastic actions than the more distant ones. This approach is typical for a human, too.

The information about positions of obstacles and the target is satisfactory to determine the control action for the vehicle. In our case there are two actions — for the left (node 13) and right (node 14) wheel of our two-wheeled car that are in form of the angular changes, which will be then transformed to powers of connected motors. In such a manner we get the basic structure of FCM for navigation as depicted in the fig. 6. Nodes 1 – 6 are for obstacles and 7 – 12 for the target. Concerning the sectors in the *visual matrix* (fig. 5 b) the nodes are labeled in the following manner: 1, 7 – *far left*; 2, 8 – *far straight*; 3, 9 – *far right*; 4, 10 – *near left*; 5, 11 – *near straight*; 6, 12 - *near right*.

A more detailed consideration leads to a fact that can simplify the structure of connection and ultimately the learning process, too. At wheeled vehicles there is a mirror symmetry. If $\mathrm{e.g.}$ an obstacle on the left causes any action then an obstacle on the right will cause the same action but with opposite direction. So for arbitrary design of the *connection matrix* E the following parameters should be in E. Only their absolute values can vary depending on the design but

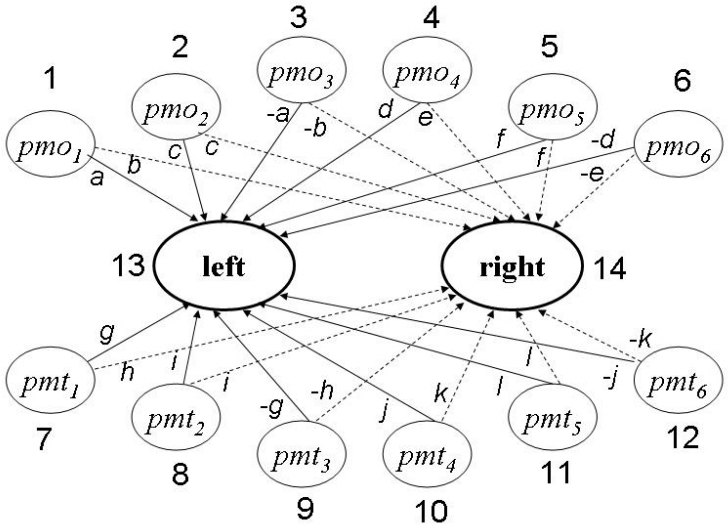


Figure 6: Structure of FCM for navigation with numbered nodes and lettered weights; solid lines for the left and dashed lines for the right wheel.

not their mutual relations (see also fig. 6):

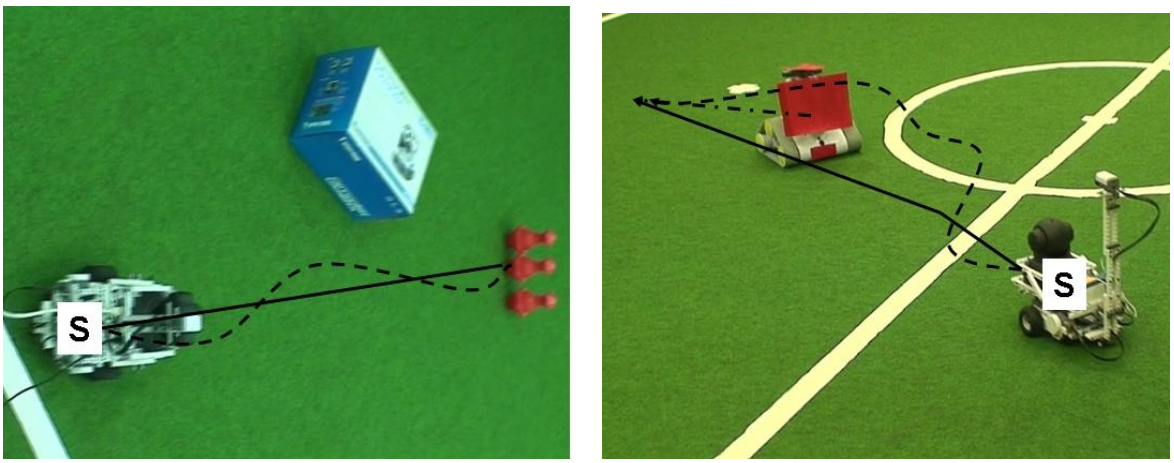
$$E^T = \begin{bmatrix} 0 & & & \cdots & & 0 & & \cdots & & 0 \\ \vdots & & & & & \vdots & & & & \vdots \\ 0 & & & \cdots & & 0 & & \cdots & & 0 \\ a & c & -a & d & f & -d & g & i & -g & j & l & -j & 0 & 0 \\ b & c & -b & e & f & -e & h & i & -h & k & l & -k & 0 & 0 \end{bmatrix}^T . \tag{4.2}$$

The navigation task is performed in the following closed loop: image sensing & processing $\rightarrow$ computation of a control action by FCM $\rightarrow$ performing the control action $\rightarrow$ sensing new image, etc. However, only a small part of this loop is described by FCM. Conventional approaches in describing a problem by FCM (see the section 2) are only once initialized by the initial node vector $C(0)$ and then we will be able to observe temporal evolution of all processes and variables in FCM. Such a system is practically closed in the face of outer impulses and it is deterministic. However, in our case the system is initialized by new sensor values in each time step and FCM needs to be open for new sensor values because we cannot precisely estimate movement of the target. In this case we speak about *FCM with active inputs* (FCM-AI) (nodes 1 – 12, see fig. 6), which are steadily fed by new values. Of course, there is usually a strong relation between individual images but because of the target's behaviour (at least) we can speak only in terms of probabilities or uncertainties and 'closed' FCM are unusable. However, beside the nodes of *active inputs* the evaluation process is the same, i.e. formula (2.3) or (2.4). *Active inputs* are evaluated by functions defined in their nodes — see the mapping α in the definition of FCM (2.1). This is a problem of the task definition and it depends on the application. In our case we used for nodes 1 – 12 the formula (4.1) and for outputs 13 and 14 the formula (2.3).

Concerning the implementation of SOMA for learning FCM the first two questions are what is represented by individuals and how they are constructed. The i-th individual is a vector $\{x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,D}\}$, which is an element of the population P, see (3.2). D as the number of optimised arguments can correspond to the number of adapted connections. Thus one individual will represent the whole design of one FCM in form of nonzero connections of the matrix E. The population will be then a set of NP concurrent and mutually competitive FCM, which in particular migration cycles exchange information about their fitness (i.e. cooperation) with the aim continuously to find the optimum solution, i.e. the best combination of connection weights. Due to the parameter AE there is a chance of obtaining a robust solution because SOMA does not prefer elitism but it tries to find proper solutions for all individuals and thereby to prevent finding although an excellent but very sensitive solution. As in the matrix E (4.2) there are only 12 nonzero connections the parameter D will be set to 12, too. Besides, still one modification was proposed. The *perturbation vector* $\vec{v}_{PRT}$ using only ones and zeros can cause too strong changes. From this reason it is constructed as $\vec{v}_{PRT,j} = \mathrm{rnd}_j$ where the function rnd is normalized to $[0; 1]$.

5 Experiments and Evaluation

For experiments our own construction of a vehicle based on Lego Mindstorms together with an IP camera was used. Because of technical limitations of both equipments it was necessary to modify experiments, too. For quality evaluation of the proposed FCM-AI two basic scenarios of experiments were proposed, see fig. 7: (a) catching static skittles and (b) catching a moving target. First, the simpler experiments with skittles were done, which are easier to be analysed and after some modifications we tried to solve catching a moving target, which was in form of another vehicle operated by a human using a joystick. Experiments with a moving target should approve the ability of the proposed method to track a ball, too.



(a) (b)

Figure 7: Experimental scenarios for static (a) and moving targets (b); solid line – optimum, dash-dot line – target trajectory and dashed line – real vehicle trajectory.

For comparison of quality of used methods first of all the time necessary to reach the target t_T

was measured. During the movement of the vehicle we can observe often oscillations around the straight line from the starting point S to the target (see fig. 7), which should be minimised. These oscillations can be represented as distance deviations $d(t)$ from the straight line (measured perpendicularly) in the time t. In the case of a moving target it is complicated to measure oscillations because the trajectory of the target is changing in time, which should be taken into account and thus the construction of the optimum trajectory would be very complicated. In the first scenario these two criteria were merged into one numerical value expressing the *error integral* ε of the surface between the straight (optimum) and the real trajectory for the time extent $t \in [0; t_T]$:

$$\varepsilon = \int_0^{t_T} |d(t)|\, dt. \tag{5.1}$$

Experiments performed in the first scenario are evaluated by ε and t_T whereas in the second scenario only by t_T. The aim is to minimise these values.

At first the matrix E is proposed manually by an expert in a common form for both scenarios with the following connection weights, see (4.2):

$$E_M = \{a, \ldots, l\} = \{-0.5,\, 0.5,\, 0.3,\, -1.0,\, 0.9,\, 0.7,\, 0.3,\, -0.3,\, 0.1,\, 0.5,\, -0.5,\, 0.5\}. \tag{5.2}$$

In this paper experiments beside our approach also for *active* HL (AHL), *nonlinear* HL (NHL) (Papageorgiou et al., 2006), SA (Ghazanfari et al., 2007) and PSO (Papageorgiou et al., 2005) were performed and mutually compared, i.e. including the manual design there are in total six approaches for constructing FCM. Experiments were repeated for all six approaches under various starting positions and configurations of objects but always with the constant initial distance to the target. Although the adaptation methods beside NHL are able basically to adapt also an initially empty matrix E (zero connections) it will be better if we can start with the initial design of E, which will be improved in next steps. For this purpose the manual design E_M is used and the task of all adaptation methods is to improve it. All weight changes were restricted only for parameters in (4.2) and no new connections were created.

Figure 8 offers a detailed overview about behaviour of individual FCM designs in the case of static targets during a typical experiment. In the sense of comparing methods by values of ε and t_T we obtained similar results in other experiments, too. There are depicted deviations of real trajectories from the optimum ones. We can observe initial oscillations for each method that are damping continuously. However, there are differences for individual methods according to deviations and time necessary to get to the target.

Although the manually designed FCM reached its target but its movement was affected strongly by oscillations. Considerable deviations caused slowing and both evaluation values were the worst. Other methods proved more or less to improve the initial FCM. AHL and NHL show similar behaviour where AHL can a little quicker reach the steady state. However, these methods exhibit a steady deviation from the optimum trajectory and therefore strictly said they missed the target (in the table 1 the quality criteria are not measurable and therefore they are indicated by –) although very closely and in the reality it is negligible. This only confirms their

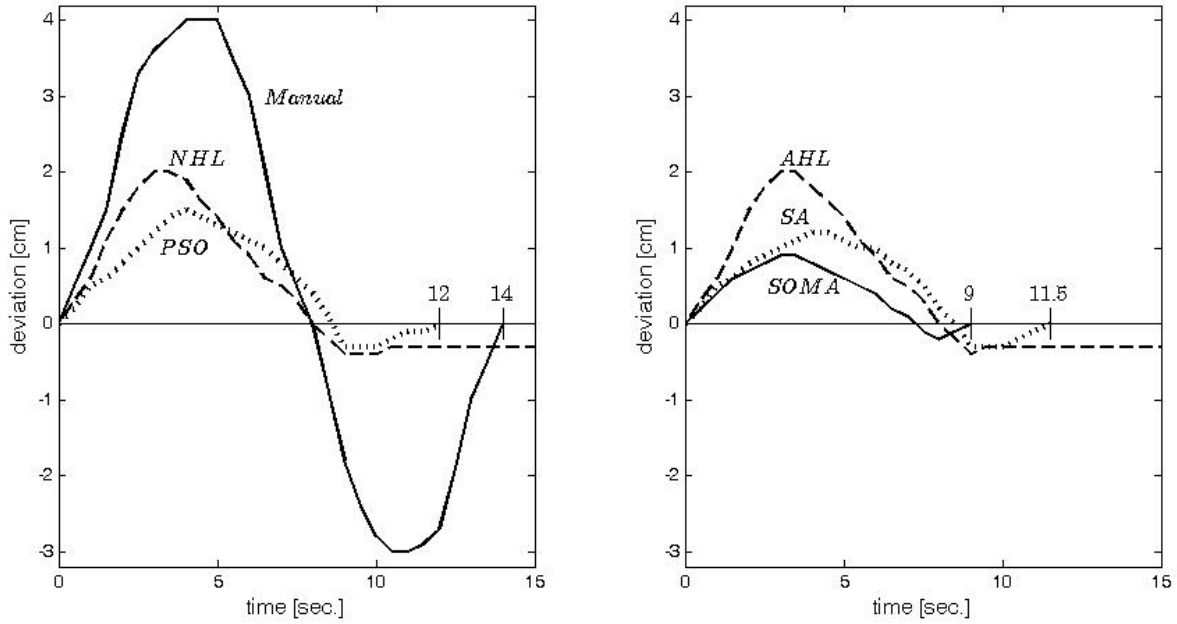


Figure 8: Position deviations of an autonomous vehicle for used FCM designs.

principal application difficulty in control. However, their convergence speed is anyway of interest. PSO and SA show unambiguously better results. They have similar behaviour but SA is slightly better. As the best solution, mainly about the *error integral*, is that provided by SOMA, which was obtained for: $Mass = 3$, $Step = 0,11$, $PRT = 0,2$, $D = 12$ (because in (4.2) there are 12 parameters), $NI = 40$, $M = 20$ and $AE = 3,5$. The final parameters of the best SOMA solution are:

$$E_{SOMA} = \{a, \ldots, l\} = \{0.97,\ -0.74,\ -0.3,\ 0.81,\ 0.45,\ 0.74,\ 0.15,\ -0.99,\ 0.68,\ 0.93,\ 0.2,\ 0.47\}. \tag{5.3}$$

If we look at fig. 8 we can see that the behaviours of all used methods resemble more or less to damped harmonic oscillations, whose behaviour $d(t)$ is similar to a curve

$$d(t) \approx \exp(i.\omega_0.\sqrt{1-\zeta}), \tag{5.4}$$

where $\omega_0 = 2\Pi.f$ is the *angular frequency*, f is the ordinary frequency and ζ is the so-called *damping ratio*, which can be also determined from the graph if we detect two successive peak positions of amplitudes x_0 and x_1, respectively

$$\zeta = \frac{1}{\sqrt{1 + (\frac{2\Pi}{\ln(x_0/x_1)})}}. \tag{5.5}$$

This approximation enables us in advance to estimate the real time, which is necessary to reach the target. As the mentioned parameters ω_0 and ζ depend on mechanical characteristics of the robot like its mass or reaction time and hence they are constant for a robot, i.e. we can

parameterize (5.4) and process it until it will be damped, which means that the target was achieved.

In the case of the moving target the times for reaching it were of course worse but beside AHL and NHL all other methods were able to fulfill the task. Just dynamic environment goes off reasonable use of methods based on Hebbian learning. Besides, we can observe the efficiency of the manual design is not so bad comparing to other designs than in the case of a static target. It means the manual design is more general and basically robust as well as it contains probably some kind of prediction knowledge about the target behaviour and its decision making (Gavalec and Mls, 2008). Table 1 shows average values of ε and t_T for the scenarios (a) and (b), respectively. The total quality of results is indirectly proportional to ε and t_T.

Table 1: Quality comparison of used methods for the criteria ε and t_T; – stands for unmeasurable.

Method	**Scenario (a)**		**Scenario (b)**
	$\boldsymbol{\varepsilon}$	$\boldsymbol{t_T}$	$\boldsymbol{t_T}$
Manual	30,55	14	24
AHL	–	–	–
NHL	–	–	–
SA	7	11,5	21
PSO	8,2	12	20,5
SOMA	4,1	9	14,5

If we compare obtained results contained in table 1 as well as depicted in fig. 8 then we can divide the used adaptation methods into three groups from the worst up to the best one: (a) AHL and NHL, (b) PSO and SA and finally (c) SOMA. For the first group the explanation for a lower quality is easy. These two methods are based on unsupervised learning, which is convenient $\mathrm{e.g.}$ for clustering but not for control. They are able to secure a steady state but it is not definite equivalent to the optimum or required one. The approaches from the second group are indeed efficient in general but PSO is very sensitive on adjusting the parameters, which is not easy and so its strength based on a large number of particles can be eliminated. On the other hand side SA is very powerful in escaping from local minima but any parallelism is missing. In other words comparing to PSO it is a one-particle algorithm because it works only with one candidate solution. It can be supposed that especially PSO and SOMA could be comparable but PSO requires complicated adjusting the parameters, which is in the case of SOMA much easier.

6 Conclusions

In this paper a new FCM design method based on SOMA was presented and on an navigation example it was compared to other known adaptation methods where it proved its quality according defined criteria as well as robustness, which can be attested by a number of various experiments. Navigation is a task, which is in principal deterministic and based on certain

rules. This fact could explain why SOMA-based adaptation shows better results than PSO, i.e. because of its more intentional (deterministic) learning. In other words, too much stochasticity in learning does not guarantee the best results in too little stochastic tasks. Therefore, it could be also very useful to focus interest on various interpolation and nonlinear methods already used in conventional rule-based fuzzy systems (Oblak, Škrjanc and Blažič, 2006; Pozna, Troester, Precup, Tar and Preitl, 2009). There is still one more aspect which should be taken into consideration. Using learning methods we can get indeed two functionally identical FCM systems but their structure is quite different. This is especially remarkable if there are no prescriptions about the structure of the FCM matrix E, which is not this case, see (4.2). Because especially FCM should reflect just human representation of knowledge. Finally, from the viewpoint of practical robotic applications we can meet with problems of lacking training data, the so-called *sparse data* and solving this problem is a great challenge for future research (Lijuan and Zhangming, 2009).

Acknowledgment

This work is the result of the project implementation: Development of the Center of Information and Communication Technologies for Knowledge Systems (ITMS project code: 26220120030) supported by the Research & Development Operational Program funded by the ERDF.

References

Aguilar, J. 2005. A survey about fuzzy cognitive maps papers, *International Journal of Computational Cognition* **3**(2): 27–33.

Axelrod, R. 1976. *Structure of Decision: The Cognitive Maps of Political Elites*, Princeton University Press.

Bertolini, M. and Bevilacqua, M. 2010. Fuzzy cognitive maps for human reliability analysis in production systems, *in* K. Cengiz and Y. Mesut (eds), *Production Engineering and Management under Fuzziness*, Vol. 252 of *Studies in Fuzziness and Soft Computing*, Springer, pp. 381—-415.

Bueno, S. and Salmeron, J. L. 2009. Benchmarking main activation functions in fuzzy cognitive maps, *Expert Systems Applications* **36**(3): 5221—-5229.

Chen, S. M. 1995. Cognitive-map-based decision analysis based on NPN logics, *Fuzzy Sets and Systems* **71**(2): 155–163.

Gavalec, M. and Mls, K. 2008. Trend evaluation in on-line decision making, *Proc. KOI 2008, Pula, Croatia*, International Conference on Operational Research, pp. 267–274.

Ghazanfari, M., Alizadeh, S., Fathian, M. and Koulouriotis, D. E. 2007. Comparing simulated annealing and genetic algorithm in learning FCM, *Applied Mathematics and Computation* **192**(1): 56–68.

Golmohammadi, S. K., Azadeh, A. and Gharehgozli, A. 2006. Action selection in robots based on learning fuzzy cognitive map, *Proc. IEEE Int. Conf. on Industrial Informatics, Singapore*, pp. 731–736.

Groumpos, P. P. 2010. Fuzzy cognitive maps: Basic theories and their application to complex systems, *in* M. Glykas (ed.), *Fuzzy Cognitive Maps*, Vol. 247 of *Studies in Fuzziness and Soft Computing*, Springer, pp. 1–23.

Johanyák, Z. C. and Kovács, S. 2006. A brief survey and comparison on various interpolation-based fuzzy reasoning methods, *Acta Polytechnica Hungarica* **3**(1): 91–105.

Kosko, B. 1986. Fuzzy cognitive maps, *International Journal of Man-Machine Studies* **24**(1): 65–75.

Koulouriotis, D. E., Diakoulakis, I. E. and Emiris, D. M. 2001. Learning fuzzy cognitive maps using evolution strategies: a novel schema for modeling and simulating high-level behavior, *Proc. of the 2001 Congress on Evolutionary Computation, Seoul*, Vol. 1, pp. 364–371.

LaValle, S. M. 2006. *Planning Algorithms*, Cambridge University Press, Cambridge, U.K. Available at: http://planning.cs.uiuc.edu/.

Li, S. J. and Shen, R. M. 2004. Fuzzy cognitive map learning based on improved nonlinear hebbian rule, *Proc. of the Third International Conference on Machine Learning and Cybernetics, Shanghai*, pp. 2301–2306.

Lijuan, Z. and Zhangming, L. 2009. Optimal selection of design schemes for a sparse distributed pile foundation based on fuzzy optimization theory, *Kybernetes* **38**(10): 1828–1834.

Nolle, L., Zelinka, I., Hopgooda, A. A. and Goodyear, A. 2005. Comparison of an self-organizing migration algorithm with simulated annealing and differential evolution for automated waveform tuning, *Advances in Engineering Software* **36**(10): 645—653.

Oblak, S., Škrjanc, I. and Blažič, S. 2006. If approximating nonlinear areas, then consider fuzzy systems, *IEEE Potentials* **25**(6): 18–23.

Papageorgiou, E. I., Parsopoulos, K. E., Stylios, C. D., Groumpos, P. P. and Vrahatis, M. N. 2005. Fuzzy cognitive maps learning using particle swarm optimization, *International Journal of Intelligent Information Systems* **25**(1): 95–121.

Papageorgiou, E. I., Stylios, C. D. and Groumpos, P. P. 2006. Unsupervised learning techniques for fine–tuning fuzzy cognitive map causal link, *Int. Journal of Human–Computer Studies* **64**(8): 727–743.

Pozna, C., Troester, F., Precup, R. E., Tar, J. K. and Preitl, S. 2009. On the design of an obstacle avoiding trajectory: Method and simulation, *Mathematics and Computers in Simulation* **79**(7): 2211–2226.

Stach, W., Kurgan, L., Pedrycz, W. and Reformat, M. 2005. Genetic learning of fuzzy cognitive maps, *Fuzzy Sets and Systems* **153**(3): 371—-401.

Vaščák, J. and Hirota, K. 2011. Integrated decision-making system for robot soccer, *Journal of Advanced Computational Intelligence and Intelligent Informatics* **15**(2): 156–163.

Vaščák, J. 2005. Evolutionary migration algorithms for scheduling, *Proc. of the 3th International Symposium on Applied Machine Intelligence and Informatics*, pp. 21–32.

Vaščák, J. and Madarász, L. 2010. Adaptation of fuzzy cognitive maps —- a comparison study, *Acta Polytechnica Hungarica* **7**(3): 109–122.

Zelinka, I. 2002. *Artificial Intelligence in Problems of Global Optimization*, BEN, Prague. in Czech.