Transactions on Information and Communications Technologies vol 16, © 1996 WIT Press, www.witpress.com, ISSN 1743-3517

Simulation of intelligent robot behavior based on reinforcement learning and neural network approach

Bojan Jerbić, Katarina Grolinger & Božo Vranješ
University of Zagreb, Faculty of Mechanical Engineering & Naval Architecture, I. Lučića 5, 10000 Zagreb, Croatia
Email: bojan.jerbic@fsb.hr

Abstract

This paper is concerned with the designing of an intelligent planning system, particularly with the planning of intelligent robot behavior in assembly processes. This comprises the robot's capability to act in unpredictable and chaotic situations, which require not just a change but the innovation of the robot's working actions. Planning of intelligent robot behavior addresses three main issues: finding task solutions in unknown situations, learning from experience and recognizing the similarity of problem paradigms. The paper presents a planning system which integrates the reinforcement learning method and a neural network approach with the aim to ensure autonomous robot behavior in unpredictable working conditions.

The assumption is that the robot is a tabula rasa and has no knowledge of the work space structure. Initially, it has just basic strategic knowledge of searching for solutions, based on random attempts, and a built-in learning system. It explores the work space by a simple sensor system, learning on-line, and being rewarded for successful action or punished for the action which does not lead to the goal state. The reinforcement learning method is used here to evaluate robot behavior and to induce new or to improve the existing knowledge. The acquired action (task) plan is stored as experience which can be used in solving future similar problems. The recognition of similar problems is established on recognition of work space structure as a structural assignment problem.

To solve the structural assignment problem using unsupervised learning, the neural network is designed following the Adaptive Resonance Theory (ART). The ART is mostly suited for one-dimensional problems or translation and rotation invariant pattern recognition. Hence, the novel concept with fuzzy learning rule and shadowed hidden layer architecture is developed. It enables the recognition of lightly translated or rotated patterns and does not forget already learned structures.

The intelligent planning system is simulated using object-oriented techniques and verified on planned and random examples, proving the main advantages of the proposed approach: autonomous learning, which is invariant with regard to the order of training samples, and enables single iteration learning progress. Simulation results also show the rigid and stable character of the proposed learning model in unpredictable examples.

1 Introduction

The robot has the main role in flexible assembly systems, but its process flexibility is not given by its own concept and must be provided by additional adequate solutions relating to adaptive robot tools, exchangeable working environment and intelligent planning system.

The paper attempts to bring into focus the intelligent robot planning problem, which includes the providing of autonomous capabilities: searching for alternative solutions in disturbed assembly processes, learning new action (task) plans and recognizing the similarity of problem paradigms. This comprises the robot's capability to act in unpredictable and chaotic situations, which require not just the change but the innovation of the robot's working actions.

In this paper the intelligent planning system integrates the reinforcement learning method and a neural network approach. The searching solution method is primarily based on general strategic procedure and random attempts. It is integrated with the reinforcement learning system which verifies and collects procedural knowledge by punishing wrong actions or rewarding successful ones. The reinforcement learning algorithm defines a behavior for the autonomous agent by maximizing total reward for possible actions that reach the goal state [1]. In order to use the learned experience the robot should be able to store not just the acquired procedural knowledge, but also the descriptive knowledge about work space structure. Besides, it should be able to recognize the similarity of a new problem with relation to a known one or to create a new problem class. Unsupervised learning by neural network appears as the efficient way for solving this structural assignment problem. Using the premises of the Adaptive Resonance Theory, the neural network is designed involving a novel concept with the fuzzy learning rule and shadowed hidden layer architecture. It overcomes the inability of the ART to efficiently deal with two-dimensional problems or translation and rotation invariant pattern recognition.

The intelligent planning system is simulated using object-oriented techniques and verified on planned and random examples, proving the main advantages of the proposed approach: autonomous learning, invariant with regard to the order of training samples, providing single iteration learning progress and keeping learned structures without forgetting the effect.

2 Planning of intelligent robot behavior

The planning of intelligent robot behavior can be seen as a kind of control system which is expected to provide the human-like and autonomous behavior of robots confronted with unknown problems. The first questions that arise are: how can we build a knowledge base about something we do not know, and how are we able then to teach the robot to cope with unknown problems? The only answer we could see is: the robot should be taught to learn on its own.

2.1 Learning approach

Knowledge is always the result of learning, whether it came from given solutions or was generated by the induction or deduction process. The intellect, when facing a certain problem, primarily generates a solution on the basis of already acquired knowledge and experience [2, 3]. Similar paradigms of shapes, situations and problems are recognized, and consequently known answers are used. This approach is based on expert knowledge, knowledge that is beyond doubt. The computer implementation of such systems is known as expert systems or knowledge-based systems, learned through a handcoded programming process. An expert system is limited by the range of domain knowledge and by the inference engine which maintains a simplified capability to synthesis and to generate new knowledge.

If the elements of a given problem have not enough similarity with existent knowledge, the knowledge about knowledge should be applied, actually the knowledge about making new science, generating new hypotheses, conclusions and solutions. This is an active learning whose computer implementation implies the normalization of scientific methods and represents a very important line of research in the development of artificial intelligence.

Obviously, the addressed problem of intelligent planning deals with an unpredictably large number of problem varieties which cannot be captured in the exact knowledge domain. Therefore, its development should be based on active learning methods.

2.2 Proposed system

The robot learning system should provide the acquisition of procedural knowledge, actually moving action rules and procedures, as well as learning descriptive knowledge on the working environment structures. The starting assumption is that the robot does not know anything. It is a tabula rasa and has just some knowledge about certain strategic general procedures for attaining objectives (grasp work-piece over the obstacle, insert misalign part and so on) and about the learning method.

Figure 1. illustrates the proposed concept of intelligent planning system. It implements the reinforcement learning [4] based on strategic and random attempts. The trial and error searching procedure converges to a given goal by punishing wrong action or rewarding successful one. After approaching the goal, reasonable actions should be synthesized and recorded as some kind of experience that would help solving enough similar problems in the future. Similar problems are those that resemble each other with regard to structure of the work space. Since the gathered knowledge consists of procedures assigned to appropriate space structures, the robot is able to follow the trained procedure, expecting to accomplish the task with a minimum number of faults.

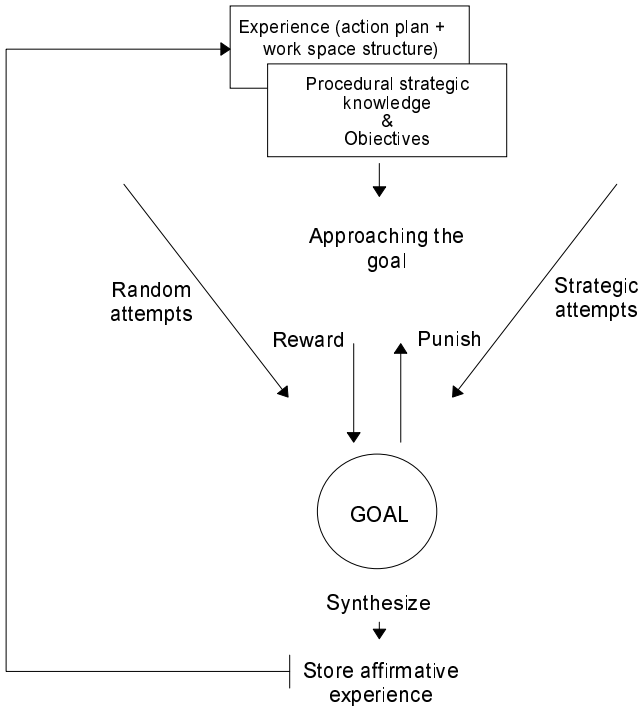


Figure 1: The concept of the robot learning system

3 Simulation model

The simulation model of the intelligent planning system includes: the robot, its work space with obstacles, and the goal (target). The model simplifies the problem, limiting robot work just to the *XZ* plane movement of its effector or to itself, with the aim of reaching the goal state $g \in S$, where S is the finite set of states $s \in S$ in the given work space or problem domain (Figure 2.).

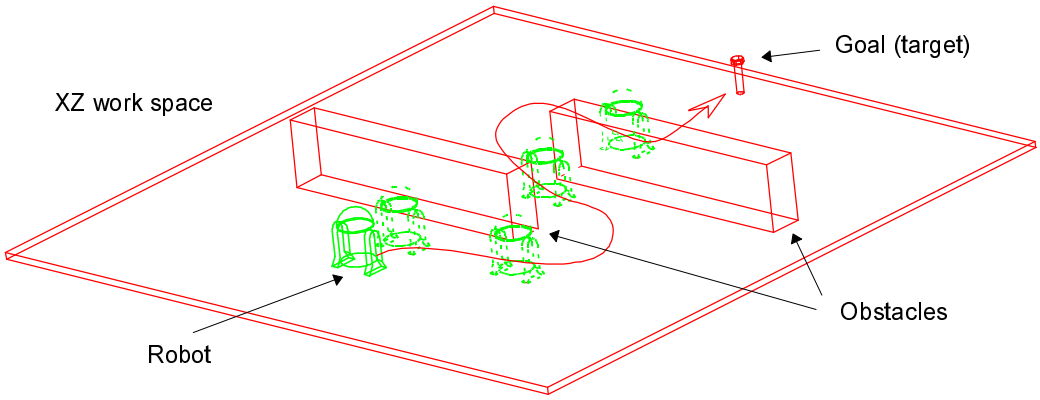


Figure 2: The graphical scene of the simulation model

Four components build up the simulation program: the graphical model of work space, procedural knowledge about the action searching strategy, the learning method and the structural assignment system.

The graphical model simulates the robot working domain using object-oriented techniques (C++ and Inventor tools [5]). It describes the geometry of simulation system elements as objects with certain characteristics and capabilities which use the learning instrument to conceive a real system.

3.1 Solution searching strategy and learning method

The central part of the solution/learning module is a simulated robot vision system (Figure 3.). It continuously gathers information on the work space from the point of the robot's or its effector's view.

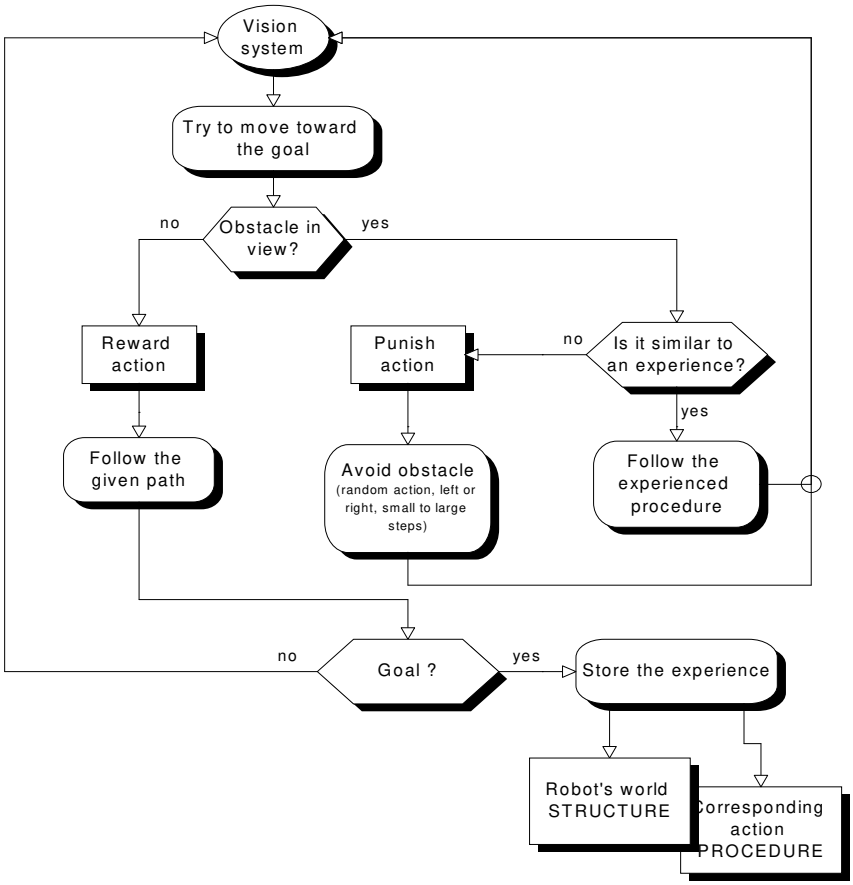


Figure 3: Solution searching strategy and learning method

The vision system oscillates inside two sensor loops looking for obstacles on the robot path. If there is no obstacle that constrains the robot from moving along a programmed or experienced path, it follows the approaching procedure.

If the vision system identifies the obstacle, the procedure is switched to the searching loop. The searching is based on random actions $a \in A(s)$. $A(s)$ is the finite set of actions that can be executed in $s \in S$: with either left, right, forward

or backward movement. The beginning action is initiated by a random generator that is connected with the local computer timer. It prevents predictable and invariant robot behavior in similar situations during the training process. Invariant behavior can result in a poor solution domain. At the beginning of the searching action the robot moves by small steps in the initiated direction: left or right. If the vision system reports that the obstacle has not been avoided, the searching strategy proposes the opposite direction: if the left-guided action was initiated, the search in the right direction follows, and vice versa. Concerning the state of the robot's vision sensors, the searching procedure continues either forward, if an obstacle has been avoided, or left/right, applying larger steps, or even backward if necessary. The searching strategy is summarized as follows:

The assumption is that the goal is in front of the robot initial position, so the robot's progress is mainly based on forward movement.

0. If (goal *g* is reached)
   then task is accomplished
   else go to 1
1. If (obstacle *B* is not on programmed/experienced path *P*)
   then {*follow_P*; go to 0}
   else go to 2
2. If (*B* is (in front) & (not in left) & (not in right)) then go to 3
   else if (*B* is (not in front) &
   (in left or in right)
   then {*moveForward(step)*; go to 0}
3. Avoiding procedure:
   generate random number *R* [0, 99]
   if (*R* > 49)
   then {*rightMove*=on; *leftMove*=off}
   else {*rightMove*=off; *leftMove*=on}
   if (*rightCycle* < 5 or *leftCycle* < 5)
   then if (*rightMove* is on)
   then if (*rightCycle* < 4)
   then {*moveRight(step); rightCycle*+1}
   else {*rightMove*=off; *leftMove*=on;
   moveLeft(4 steps); rightCycle+1}
   else if (*leftMove* is on)
   then if (*leftCycle* < 4)
   then {*moveLeft(step); leftCycle*+1}
   else {*rightMove*=on; *leftMove*=off;
   moveRight(4 steps); leftCycle+1}
   else {*rightCycle*=0; *leftCycle*=0}
   go to 0

When the obstacle remains behind the robot, it should try to return to the programmed path, switching back to approaching procedure.

The reinforcement learning algorithm rewards each action $a \in A(s)$ with immediate reward $r(s, a)$, $r \in [-1, 1]$, which is positive if the action a results in the new state $s(t+1) \in S$ that avoids the obstacle and enables the robot to approach the goal state $g \in S$, or negative, if the new state $s(t+1)$ limits the movement action of the robot. The positive actions, which guide the robot to the goal, are learned as an action plan accompanied with related work space structure, building the robot's experience. The action plan $Pl(\ lm(d), rm(d), fm(d), bm(d)\ )$ is stored in a list which contains the move sequence as the function of the move distance d, where $lm(d), rm(d), fm(d), bm(d)$ are left, right, forward and backward actions respectively.

Since the discovered action plan follows the particular structure of work space, any similar structure encountered in the future will be solved by the same plan. *It means that learning is basically achieved in one iteration.* The variable values are just the length of the movement d, which could vary depending on particular obstacle position and dimension. In the following training or working phase, when the robot recognizes that the actual situation is similar to the experienced one and the action plan is used as a paradigm to help reaching the goal in a new task, the reinforcement learning system ensures the improvement of behavior knowledge by minimizing total reward:

$$U_j = \sum_i c(d)_i \cdot r(s,a)_i \ ; \ i = 1, 2, ..., n \tag{1}$$

where U_j is the total reward for plan Pl_j,
n is the number of actions in the plan Pl_j.

$$c(d) = \begin{cases} d(t+1) > d \Rightarrow d = d + \Delta \\ \\ d(t+1) < d \Rightarrow d = d - \Delta \end{cases} \tag{2}$$

$d(t+1) = f(s(t+1)),\ s(t+1) \in S$

where Δ is the length learning parameter.

In this way the learning process never stops, providing task-oriented control of robot behavior, minimizing trial and error efforts and maximizing overall robot performance efficiency.

3.2 Structural assignment by neural network

The assigned work space structure cannot be simply recorded as a symbolically coded description, because this would require a large computer memory for any new or similar situation encountered. Consequently, it would take a considerable time for searching so large a data base. The neural network model offers an efficient way to process a large number of input values and to memorize them condensed as synaptic weights. Applied to the structural

assignment problem, the neural network should be designed to memorize distinct and to recognize similar work space structures. The notion "similar situations" denotes here situations that are usually considered similar, i.e. situations where obstacles form similar spatial relations, lightly translated or rotated in any direction, or where they slightly differ in shape. The work space structure is represented as the $n \times m$ binary matrix, which is assumed to be a network input vector. The zeros represent empty space and units denote the discrete space where the obstacle resides.

3.2.1 Related works

The Adaptive Resonance Theory (ART) is the most common approach in unsupervised learning problems [6]. The ART is very successful when applied to cell formation problems in group technology and generally to one-dimensional problems or translation/rotational invariant pattern recognition. For this reason, the application of the ART to structure assignment problems shows some weaknesses:

- The order of training samples can significantly affect network performance.
- Matrix density (proportion of "1" elements) influences network efficiency. If matrix density is higher, similarity recognition is better. If the matrix's density tends to be low, it is advantageous to reverse the units and zeros in input matrix.
- Once a new category is learned, the weights can be gradually modified in the subsequent training period by slightly varied new patterns. But, the modification can be so substantial that the network is eventually unable to recognize the original pattern - the network is forgetting.
- Inability to classify translated or rotated patterns/obstacles.
- The number of categories tends to become unacceptably large.

Similar problems have been encountered in the application of fuzzy ART [7], which is modified ART1. The fuzzy ART network uses a new learning law, and, in addition to the similarity coefficient (ρ) a learning rate parameter (β) is involved. With the new learning law, the neural network does not forget the already learned situations. Because of a usually low β, the network learns slowly and it can occur that even very distinct structures fall into the same group. This can be avoided by repeating the learning cycle a few times, but this prolongs the learning period. In general, fuzzy ART suffers from the same weaknesses as the original ART1 network, except for the already mentioned forgetting aspect.

3.2.2 Modified fuzzy ART with shadowed hidden layer

In order to provide recognition of work space structures according to the given definition of similarity and without the previous limitations, the modified fuzzy ART neural network is designed. It divides the learning process into three phases: initiation, training and working.

In the primary phase it uses the Cluster Center Seeking approach (CCS) to prevent the blending of distinct structure categories, provide their balanced dissipation and eliminate the influence of the order of problem presentation. The algorithm does not completely follow the CCS algorithm known from [8]. Instead of calculating the maximum absolute dissimilarity - minimum absolute similarity, maximum relative dissimilarity - minimum relative similarity S_p is calculated. The influence of the size of the obstacle on the cluster center identification is thus included, considering the relation between the matching level (number of matching "1's") and obstacle size (number of "1's"). The cluster center patterns actually represent the most distinctive structure samples that must initiate the network. The number of identified cluster centers must not correspond to the number of categories on the output layer of neurons, since the network can evaluate high similarities for some centers, which then trigger the same output neuron.

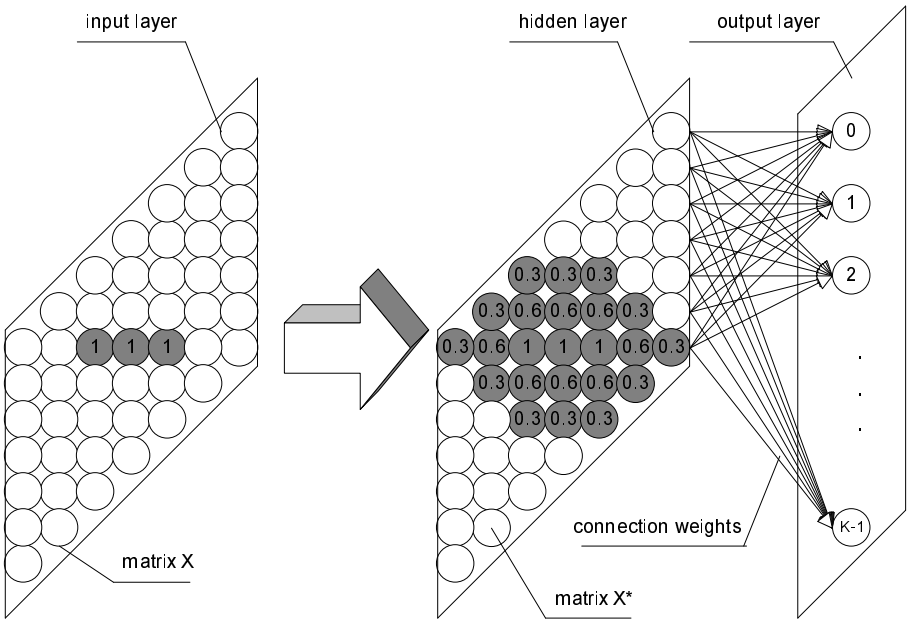


Figure 4: Modified fuzzy ART with shadowed hidden layer

The network architecture is based on three neuron layers. The purpose of the hidden layer is to modify the input vector in order to provide the recognition capability of bit-translated or rotated situations (in any direction) as similar situations (triggering the same output neuron). It is attained by adding shadows around obstacle contours. Shadow intensity decreases as the distance from the obstacle increases two or more levels deep. In this way the input layer is transformed from binary to analogue vector as can bee seen in Figure 4. Each neuron in the hidden layer is connected to every neuron in the output layer. After the network is initialized by the distinctive structure samples, the

secondary learning-training phase follows, including all of the structure samples. The steps involved in the network algorithm are stated below:

1. Initialization:
β ($0 < \beta < 1$) - learning rate parameter,
ρ ($0 < \rho < 1$) - similarity coefficient,
$N = 0$ - number of neurons in the output layer.
C_{max} - maximum number of cluster centers.
2. Read all input matrices X which consist of binary elements.
3. Find the matrix with minimum density dn that will be the first cluster center:

$$dn = \min_{p} \left\{ \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} x_{p_{ij}}, p = 0,1,..,(Np-1) \right\}, \tag{3}$$

where Np is the number of input matrices, $x_{p_{ij}}$ is ij element of p-th input matrix X_p, n is the number of rows in the input matrix, m is the number of columns in the input matrix.
Matrix X_d ($d \in \{ 0, 1, ..., (Np-1) \}$), with maximum density dn becomes the first cluster center matrix $Y_1 = X_d$ ($C = 1$, C - the number of already existing centers).
4. Transform the input matrix including the shadows $X \rightarrow X^*$
X - input binary matrix, X^* - shadowed analogue matrix.
5. Calculate the relative similarity S_p between existing centers/center and the rest of matrices:

$$S_p = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \sum_{l=0}^{C} \left(x^*_{p_{ij}} \cdot y_{l_{ij}} \right)}{C \cdot \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} x^*_{p_{ij}}}, \tag{4}$$

$p = 0, 1, ..., (Np - 1); X^*_p \neq Y_l \ \forall \ l$

where: S_p - similarity of p-th matrix with already existing centers,
y_{ij} - ij element of l-th matrix center Y_l,
6. Find situation with minimum relative similarity S_{min} to existing centers:

$$S_{min} = \min_{p} \left\{ S_p, p = 0,1,\ldots,(Np-1) \right\}. \tag{5}$$

The number of already existing centers becomes
$C(t) = C(t-1) + 1$, and the matrix X_p with minimum relative similarity to existing centers becomes a new center $Y_C = X^*_p$.
If the number of already existing centers is smaller than the number of needed centers (C_{max}) then go back to 5, else go to 7.
7. Apply cluster center matrix $X^* = Y_l$ ($l = 0, 1, ..., C_{max}-1$)

if $N = 0$ go to 12
else go to 9.
8. Apply next input matrix
$X^* = X_p$ $(p = 0, 1, ..., Np-1)$.
9. Compute choice (similarity) functions T_k for every output neuron:

$$T_k = \frac{\left\| X^* \wedge W_k \right\|}{\left\| X^* \vee W_k \right\|} \tag{6}$$

$k = 0, 1, ..., (N(t) - 1)$,
where: $\wedge$ is the fuzzy "and" operator: $(a \wedge b) = min\ (a,b)$,
$\vee$ is the fuzzy "or" operator $(a \vee b) = max\ (a, b)$,

$$\left\| X^* \wedge W_k \right\| = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} min(x^*_{ij}, w_{k_{ij}}), \tag{7}$$

$$\left\| X^* \vee W_k \right\| = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} max(x^*_{ij}, w_{k_{ij}}). \tag{8}$$

10. Select best matching exemplar T_θ, it is a neuron in the output layer with the largest value of choice function:

$$T_\theta = \max_k \{T_k\}. \tag{9}$$

11. If $T_k \geq \rho$ trigger neuron θ and go to 13,
if not, continue with 12.
12. Create a new output neuron (a new class) θ. Set the number of neurons in the output layer $N(t) = N(t-1) + 1$.
13. Update best matching exemplar:
a. If neuron is triggered for the first time:

$$W_\theta(t) = X^*, \tag{10}$$

W - connection weight matrix.
b. If neuron has already been triggered

$$\text{if } x^*_{ij} < w_{\theta_{ij}}(t-1) \Rightarrow w_{\theta_{ij}}(t) = (1-\beta)\cdot(w_{\theta_{ij}}(t-1) - x^*_{ij}), \tag{11}$$

$$\text{if } x^*_{ij} > w_{\theta_{ij}}(t-1) \text{ and } w_{\theta_{ij}}(t-1) > 0 \Rightarrow w_{\theta_{ij}}(t) = (1+\beta)\cdot(x^*_{ij} - w_{\theta_{ij}}(t-1)), \tag{12}$$

$$\text{if } x^*_{ij} > w_{\theta_{ij}}(t-1) \text{ and } w_{\theta_{ij}}(t-1) = 0 \Rightarrow w_{\theta_{ij}}(t) = \beta \cdot x^*_{ij}, \tag{13}$$

$$\text{if } x^*_{ij} = w_{\theta_{ij}}(t-1) \Rightarrow w_{\theta_{ij}}(t) = w_{\theta_{ij}}(t-1). \tag{14}$$

14. If all cluster centers are applied $(l = C_{max}-1)$ go to 15, else go to 7.
15. If all input matrices are applied $(p = Np-1)$ go to 16, else go to 8.
16. The end.

If the output neuron is triggered for the first time, the best matching exemplar is updated the same as in ART1. If the output neuron has already been triggered, the best matching exemplar is updated similarly to the fuzzy

ART algorithm. In the fuzzy ART network the connection weights can only decrease or remain the same. In the modified fuzzy ART algorithm connection weights are decreased or increased proportionally to the difference between values in the connection weight matrix and the modified input matrix from the hidden layer. The learning rate parameter (β) determines how much connection weights are changed in one step, that is - how fast the neural network learns.

The modified fuzzy ART with shadowed hidden layer is designed to learn all the time even in the working mode. This is needed because the assumption is that unpredictable situations which were not captured in the training phase can occur. Learning in the working mode can induce the creation of new output neurons (classes), or the improvement of existing network knowledge.

4 Simulation results

The verification of the presented planning system is carried out on two example sets. The first one includes 134 training samples constructed on the basis of six main structure paradigms. The basic structures are varied regarding to the position and dimension of obstacles. This example set was used to train a robot with no experience. The second example set includes 20 randomly generated structures and it was used to verify the efficiency of robot behavior using the knowledge learned from the previous training set.

4.1 Training results

Figure 5. illustrates six basic structures which were used to construct the training example set. The training was analyzed in two runs using the organized and random sequence of the examples.

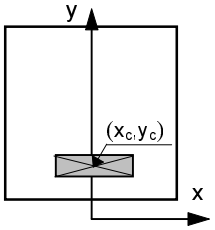
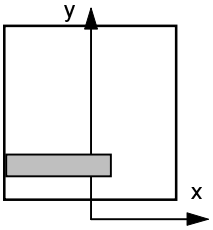
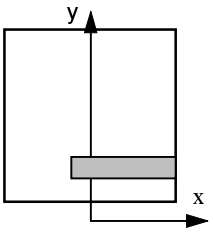
The simulation took about two hours on a 135 MHz INDY workstation with 64 MB of RAM. The time of simulation is directly affected by animation speed. Finally, the robot generated 15 paradigm classes in both runs, what proves that the order of training samples does not affect recognition capability. The training results can be summarized by the robot's knowledge level k which expresses the relative variation in path length:

$$k = \sum_{i=0}^{i=N-1} k_i \tag{15}$$

N - the number of neurons in output layer.

$$k_i = \frac{\overline{d_p}}{d_p} \tag{16}$$

k_i - the knowledge level of the i-th solution set where the p belongs,
d_p - path length for the structure p,
$\overline{d_p}$ - average path length for the i-th solution set.

BASIC STRUCTURE WITH ONE OBSTACLE						
	STRUCTURE 1		STRUCTURE 2		STRUCTURE 3	
						
BASIC OBSTACLE POSITION						
	x_c	z_c	x_c	z_c	x_c	z_c
pos. 1	0	6	-6	6	6	6
pos. 2	0	13	-6	13	6	13
pos. 3	0	19	-6	19	6	19
TRANSLATION FROM BASIC POSITION						
	x	z	x	z	x	z
trans. 1	+1	0	+1	0	-1	0
trans. 2	-1	0	0	+1	0	+1
trans. 3	0	+1	0	-1	0	-1
trans. 4	0	-1				

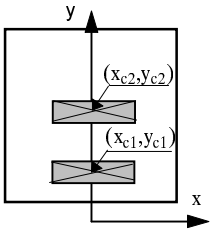
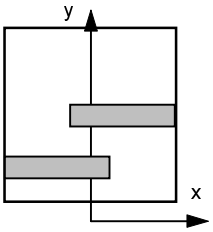
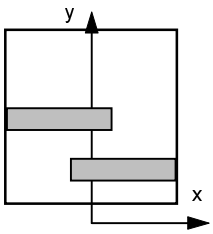
BASIC STRUCTURE WITH TWO OBSTACLES												
	STRUCTURE 3				STRUCTURE 4				STRUCTURE 4			
												
BASIC OBSTACLE POSITION												
	x_{c1}	z_{c1}	x_{c2}	z_{c2}	x_{c1}	z_{c1}	x_{c2}	z_{c2}	x_{c1}	z_{c1}	x_{c2}	z_{c2}
pos. 1	0	5	0	15	-6,5	5	6.5	15	6,5	5	-6,5	15
pos. 2	0	11	0	21	-6,5	11	6.5	21	6,5	11	-6,5	21
TRANSLATION FROM BASIC POSITION												
	x_1	z_1	x_2	z_2	x_1	z_1	x_2	z_2	x_1	z_1	x_2	z_2
trans. 1	+1	0	+1	0	+1	0	-1	0	-1	0	+1	0
trans. 2	-1	0	-1	0	0	+1	0	+1	0	+1	0	+1
trans. 3	0	+1	0	+1	+1	+1	-1	+1	-1	+1	+1	+1
trans. 4	+1	+1	+1	+1								
trans. 5	-1	+1	-1	+1								

Figure 5: Basic structures and their variations.

The change of knowledge level *k*, obtained with organized sample sequence, and described in Figure 6., shows rapid increases at the beginning of simulation, as well as in some training stages when new paradigms appeared. It is the effect of learning in one iteration. After the robot has learned a certain paradigm, it can continue improving its knowledge by movement length correction. It is reflected as a slight but stable oscillation of the knowledge level curve. When the curve reaches overall stability it can be concluded that the robot has achieved a competent level of knowledge for a given finite problem domain.

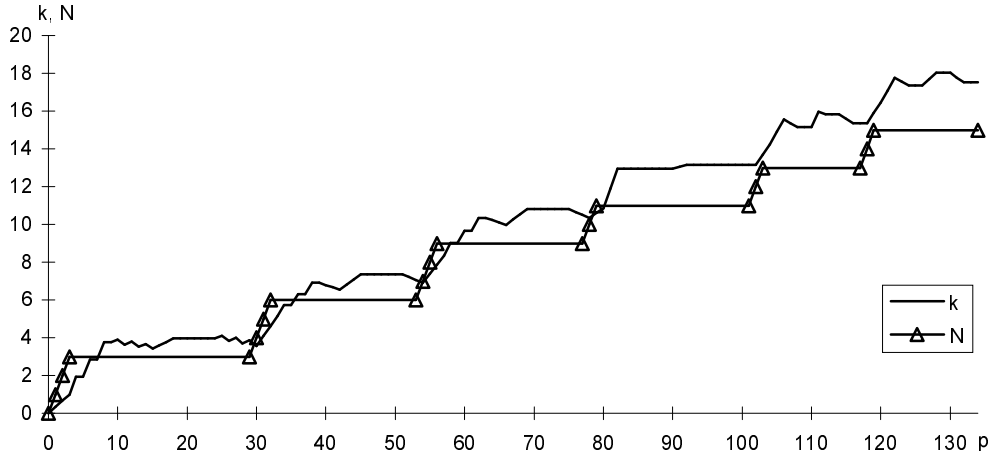


Figure 6: The knowledge level (*k*) and number of output neurons (*N*) during the training with the organized sequence of samples.

The training which applied the random sequence of samples gave even better results as can be seen in Figure 7. The knowledge level increases rapidly at the beginning of training, reaching the maximum by 65 samples, and keeps it stable until the end of run.

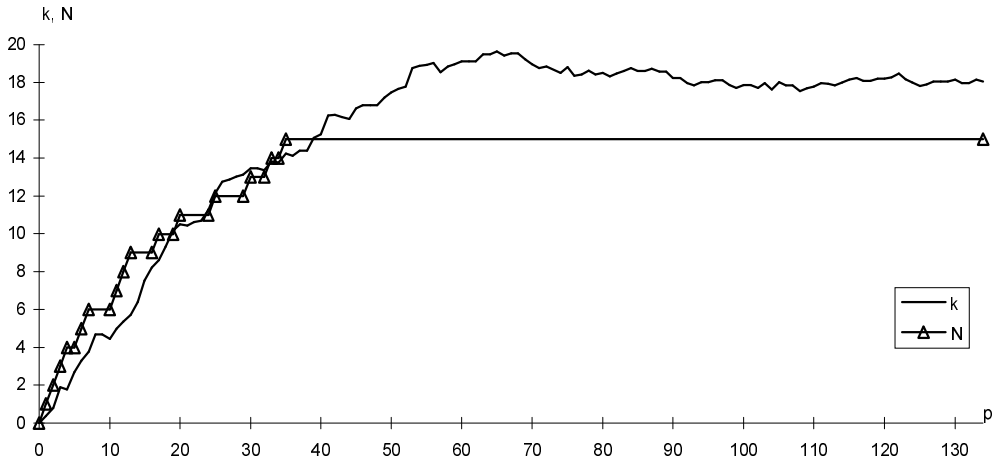


Figure 7: The knowledge level (*k*) and number of output neurons (*N*) during the training with the random sequence of samples.

4.2 The behavior of the experienced robot

The knowledge from the previous training phase was used to solve the example set which was randomly generated, varying obstacle dimensions and positions without any scheme. As a result the considerably different structures were obtained. However, the curve of knowledge level in Figure 8. clearly shows the stable robot's proficiency, since there are no many curve leaps. Only three new classes have been generated. All other examples have been solved using the robot's training experience, without wasting time by searching for a way out.

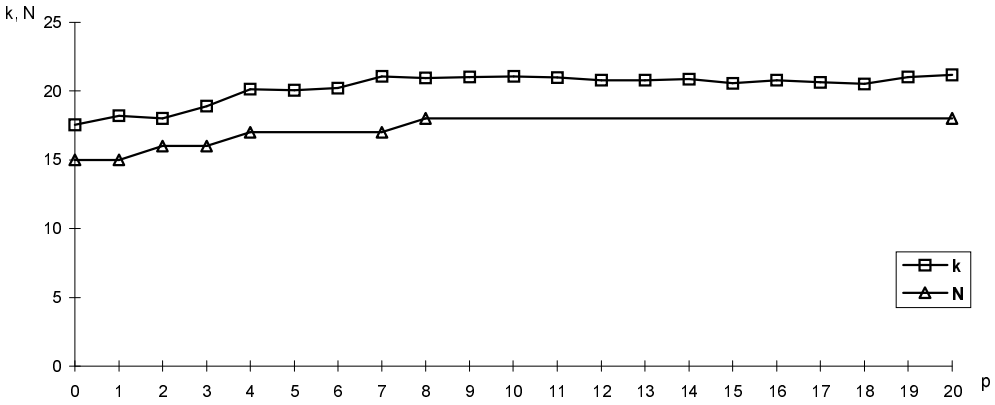


Figure 8: The knowledge level (k) and number of output neurons (N) on the random examples using the experience.

5 Conclusions

The paper has introduced a learning system which attempts to produce intelligent robot behavior in unpredictable and chaotic situations. The system employs the reinforcement learning method for acquiring new solutions and a neural network, based on Adaptive Resonance Theory, for the recognition of work space structure similarity. The searching method enables solving unknown problems, following general and random procedures, as well as similar problems, using acquired knowledge. The reinforcement learning method is used to evaluate robot behavior and to induce new or to improve the existing knowledge. The neural network is applied as a kind of robot visual memory, and its architecture is realized in order to contribute to the recognition of complex spatial structures. It has been attained by introducing the shadowed hidden layer and a new learning rule.

The modified fuzzy ART with shadowed hidden layer showed the following advantages:

- Slightly translated or rotated patterns can be recognized as similar.
- The neural network does not forget already learned situations.
- The results are not affected by the order of problem presentation.

- Input matrix density does not affect the network performance.

The presented planning system is verified on planned and random example sets. Simulation results prove the robot's capability to learn behavior in one iteration, showing the rigid and stable character of the proposed learning model.

Acknowledgment

This work is part of the research project "Optimizing integrated production", supported by the Ministry of Science and Technology of the Republic of Croatia.

6 References

1. Koenig, S. & Simmons, R.G. *Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains*, Technical Report CMU-CS-93-106, School of Computer Science, Carnegie Mellon University, 1992.
2. Reich, Z. The development of Bridger: A methodological study of research on the use of machine learning in design, *Artificial Intelligence in Engineering*, v. 8, n. 3, 1993., p. 217-231.
3. Jerbić, B. *Reasoning geometry by computer*, Croatian academy of science and art, Zagreb 1995.
4. Mahadevan, S. & Connell, J. Automatic programming of behavior-based robots using reinforcement learning, *Artificial Intelligence*, 1992, n. 55, p. 311-365.
5. -, *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor*, Release 2, Silicon Graphics, 1995.
6. Chen, S.J. & Sheng, C., S. A neural network-based cell formation algorithm in cellular manufacturing, *International Journal of Production Research*, 1995, v. 33, n. 2, p. 293-318.
7. Suresh, N.C. & Kaparthi, S. Performance of fuzzy ART neural network for group technology cell formation, *International Journal of Production Research*, 1994, v. 32, n. 7, p. 1693-1713.
8. Rao, H.A. & Gu, P. A multi-constraint network for the pragmatic design of cellular manufacturing system, *International Journal of Production Research*, 1995, v. 33, n. 4, p. 1049-1070.