Hierarchical fuzzy control for the inverted pendulum over the set of initial conditions

Juliusz Zajaczkowski and Brijesh Verma

School of Computing Sciences, Faculty of Business and Informatics
Central Queensland University
Rockhampton, QLD 4702, Australia
jmzaj@yahoo.com, b.verma@cqu.edu.au

Abstract. We examine hierarchical fuzzy control of the inverted pendulum over the set of initial conditions. A new compositional method for the inverted pendulum system is introduced. Three layered hierarchical fuzzy logic topology is used to create a fuzzy rule base for the control system. Fuzzy rules are learnt by evolutionary algorithm designed for the compositional method.

1 Introduction

In this paper we investigate the compositional method applied to a typical dynamical system. The inverted pendulum system provides a good test platform used for the evaluation and comparison of various control theories. The control of the inverted pendulum has been undertaken using linear and nonlinear dynamics and include both classical and fuzzy logic control techniques, to mention a few, [1],[3],[5]– [10].

The major advantage of compositional method is that once the fuzzy rule base is determined it can control the dynamical system from a range of initial conditions without additional computations. In other words, the controller is fixed and is expected to successfully control the system from a large range of initial conditions.

A first step in the construction of a fuzzy logic system is to determine which variables are fundamentally important. In considering a single layered fuzzy system, with all variables as input into this layer, the total number of rules in a system is an exponential function of the number of system variables. This 'curse of dimensionality' can be handled in a variety of ways, one being the grouping of fuzzy rules into prioritised levels to design hierarchical structure.

In a hierarchical fuzzy logic structure (HFS), typically the most influential parameters are chosen as the system variables in the first level, the next most important parameters are chosen as the system variables in the second level, and so on. The number of rules in a complete rule set is reduced to a linear function of the number of variables. The decomposition into hierarchical fuzzy logic sub-systems reduces greatly the number of fuzzy rules to be defined and to be learnt. The task of finding the fuzzy rules in hierarchical fuzzy structure is given to evolutionary algorithm (EA).

The paper has been divided into six sections. The background of this research is given in Section 1. Section 2 describes the dynamical system under investigation. Section 3 introduces the concept of hierarchical fuzzy systems and their application in

control problem of the inverted pendulum system. In section 4 we give a detailed description of the evolutionary algorithm designed to find the fuzzy rule base. Details of computer simulations are given in Section 5. Final conclusions are drawn in Section 6.

2 Inverted pendulum system

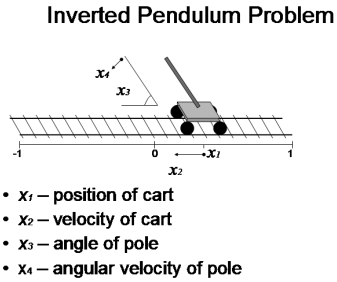

Fig. 1. Inverted Pendulum System.

The nonlinear system to be controlled consists of the cart and a rigid pole hinged to the top of the cart. The cart is free to move left or right on a straight bounded track and the pole can swing in the vertical plane determined by the track. It is modelled by [11]:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= u + m\ell(\sin(x_3)x_4^2 - \dot{x}_4\cos(x_3))/(M+m) \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \frac{g\sin(x_3) + \cos(x_3)(u - m\ell x_4^2\sin(x_3))/(M+m)}{\ell(4/3 - m\cos(x_3)^2/(M+m))}
\end{aligned}$$

where x_1 is the position of the cart, x_2 is the velocity of the cart, x_3 is the angle of the pole, x_4 is the angular velocity of the pole, u is the control force on the cart, m is the mass of the pole, M is the mass of the cart, ℓ is the length of the pole, and g is gravitational acceleration. The control force is applied to the cart to prevent the pole from falling while keeping the cart within the specified bounds on the track. We take $m = 0.1kg$, $M = 1kg$, $\ell = 0.5m$, $g = 9.81ms^{-2}$, with state limits: $-1.0 \leq x_1 \leq 1.0$ and $-\pi/6 \leq x_3 \leq \pi/6$.

The goal is to determine fuzzy controllers necessary to stabilise the system about the unstable reference position $x = 0$ as quickly as possible, whilst maintaining the system within the the target region ($\widetilde{\text{TR}}$) defined by the following bounds: $|x_1| \leq 0.1$, $|x_2| \leq 0.1$, $|x_3| \leq \pi/24$, $|x_4| \leq 3.0$. The desired fuzzy controller is required to control the system such that the state variables converge to TR and are maintained within TR for a prescribed time limit T_f with $T_f = 20.0$.

3 Hierarchical fuzzy system

Our hierarchical fuzzy logic structure has two input variables in the first layer. Then there is one input variable in second and third layer of 3-layered HFS. This input configuration provides the minimal number of fuzzy rules in the knowledge base [12].

Investigation of different topologies for the inverted pendulum system showed that 3-layered topology with angular speed x_3 and angular position x_4 as input variables in the first layer, and then the cart's speed x_2 as input in second layer and position displacement x_1 as input variable in the third layer was a best choice for the control system (in terms of state variables convergence and control magnitude) [13].

The output variable u is calculated using the Mamdani product inference engine or minimum inference engine. Given a fuzzy rule base with M rules and n antecedent variables, a fuzzy controller as given in Equation 1 (with Mamdani inference engine) or Equation 2 (with minimum inference engine) uses a singleton fuzzifier, Mamdani product or minimum inference engine and centre average defuzzifier to determine output variables.

$$u = \frac{\sum_{\ell=1}^{M} \overline{u}^{\ell} (\prod_{i=1}^{n} \mu_{A_i^{\ell}}(x_i))}{\sum_{\ell=1}^{M} (\prod_{i=1}^{n} \mu_{A_i^{\ell}}(x_i))} \tag{1}$$

where $\overline{u}^{\ell}$ are centres of the output sets B^{ℓ}.

$$u = \frac{\sum_{\ell=1}^{M} \overline{u}^{\ell} (min_{i=1}^{n} \mu_{A_i^{\ell}}(x_i))}{\sum_{\ell=1}^{M} (min_{i=1}^{n} \mu_{A_i^{\ell}}(x_i))} \tag{2}$$

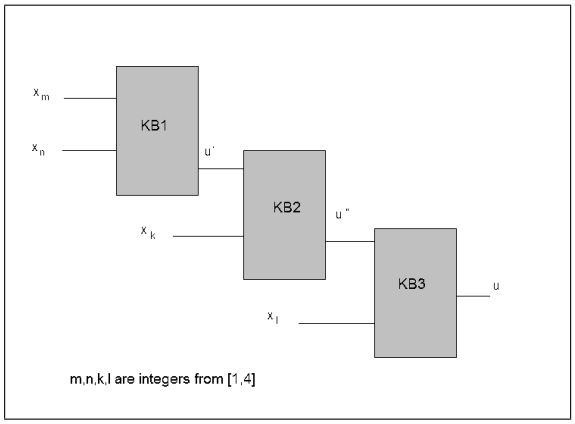

Fig. 2. Three Layer HFS

The architecture of 3-layered HFS is shown in Figure 2. For this system the first knowledge base KB_1 has the two inputs, x_i and x_j to produce as output a first approximation to the control u'. This u' together with x_k are used as input in the second

knowledge base KB_2. Then the second layer produces another approximation of control u'' which with x_l is used as input to the third (and final) layer to produce the final control output u.

In the first layer there are $25 = (5)^2$ rules in the knowledge base. In general, we may write the ℓ^{th} fuzzy rule has the form:

$$\text{If } (x_i \text{ is } A_i^\ell) \text{ and } (x_j \text{ is } A_j^\ell) \text{ Then } (u' \text{ is } B^\ell).$$

where $A_k^\ell, k = 1, 2, 3, 4$ are normalised fuzzy sets for input variables $x_k, k = 1, 2, 3, 4$, respectively, and where B^ℓ are normalised fuzzy sets for output variable u'.

For the second layer there are $35 = 7 \times 5$ rules in the knowledge base and we may write the ℓ^{th} fuzzy rule has the form:

$$\text{If } (u' \text{ is } C^\ell) \text{ and } (x_k \text{ is } A_k^\ell) \text{) Then } (u'' \text{ is } B^\ell).$$

where C^ℓ are normalised fuzzy sets for the input control variable u.

Similarly, there are 35 rules in the third layer and the ℓ^{th} fuzzy rule has the form:

$$\text{If } (u'' \text{ is } C^\ell) \text{ and } (x_l \text{ is } A_l^\ell) \text{ Then } (u \text{ is } B^\ell).$$

There are a total of 95 fuzzy rules in this hierarchical structure. The output for each layer is obtained using the either Mamdani inference engine or minimum inference engine as given in Equations 1 and 2, with the appropriate change of variable and associated membership functions for that variable.

Each domain region for x_i is divided into 5 overlapping intervals and each assigned membership sets: $A_i^k, k = 1, \cdots, 5$, which we encode numerically as integers from 1 to 5 respectively. We found that the set of 5 membership functions provides relatively small knowledge base while maintaining a good controller performance. As the output variable range was found larger (by experiments), we divided the output u into 7 overlapping regions with 7 membership sets $B^k, k = 1, \cdots, 7$, with integer encoding 1 to 7. All fuzzy membership functions are assumed to be Gaussian with their centres evenly spaced over the range of input and output variables.

4 Evolutionary algorithm for compositional method

The compositional method uses the evolutionary algorithm to search for a fuzzy rule base capable of controlling the system over the whole set of initial conditions and fitness of each individual in the EA population reflects the controller performance for every initial condition in the set. In other words, every string in the population is assigned the fitness value which is a composite value representing string's performance for every single initial condition.

The evolutionary algorithm is a heuristic search technique that maintains a population of individuals $P(t)$ at iteration t to the next iteration $t + 1$ [14]. Each individual represents a potential solution to a given problem. Each individual is assigned a measure of fitness which defines how accurate it is as a potential solution to the problem. Depending on whether the problem is defined as a maximisation or minimisation problem, the best solution may be that string with the highest or lowest fitness value. The inverted pendulum problem is defined as minimisation problem.

Each individual string in the evolutionary population is to uniquely represent the hierarchical structure. This is achieved as follows. In the knowledge base of any layer, assuming a complete and consistent knowledge base, each fuzzy rule is uniquely defined by the consequent part.

This consequent part is identified by a particular output fuzzy set, for example, B^k. Such a fuzzy set can be identified by the integer $k \in [1, 7]$. The three fuzzy rule base structure can be represented as a linear individual string of $M = 25 + 35 + 35 = 95$ consequents, $\underset{\sim}{p}_k = [a_1, \cdots, a_{95}]$, where a_j is an integer $\in [1, 7]$ for $j = 1, \cdots, 95$.

The fitness f_k of a given string for a single initial condition is evaluated as follows: given an initial condition of the system we can decode each string $\underset{\sim}{p}_k$ into the two or more components defining the fuzzy knowledge base for each layer, then use the Mamdani or minimum inference formula to evaluate u',u'', and u to find the final control to be applied at each value of the state $\underset{\sim}{x}$. Given an initial state the system state equations are integrated by the Runge-Kutta algorithm (RK4) with step size 0.02 over a sufficiently long time interval $[0, T]$. The fitness f_k to be minimised, is then calculated based on some measures of the behaviour of the system over the time interval. These include, the accumulated sum of normalised absolute deviations of x_1 and x_3 from zero, the average deviation from vertical, the average deviation from the origin or $T - T_S$ where T_S, the survival time, is taken to mean the total time before the pole and cart break some bounds. A penalty of 1000 is added to the objective if the final state breaks the following bounds $|x_1| \leq 0.1$, $|x_2| \leq 0.1$, $|x_3| \leq \pi/24$, $|x_4| \leq 3.0$, i.e., leaves the designated TR (target region).

The fitness function f_k has the general form:

$$f_k = \omega_1 F_1 + \omega_2 F_2 + \omega_3 F_3 + \omega_4 F_4 + \omega_5 F_5 \quad \text{with}$$

$$F_1 = \frac{1}{N} \sum_1^N \frac{|x_1|}{x_{max}}, \quad F_2 = \frac{1}{N} \sum_1^N \frac{|x_2|}{\dot{x}_{max}}, \quad F_3 = \frac{1}{N} \sum_1^N \frac{|x_3|}{\theta_{max}},$$

$$F_4 = \frac{1}{N} \sum_1^N \frac{|x_4|}{\dot{\theta}_{max}}, \quad \text{and} \quad F_5 = \frac{1}{T}(T - T_S),$$

where $x_{max} = 1.0$, $\theta_{Max} = \pi/6$, $\dot{x}_{max} = 1.0$, $\dot{\theta}_{Max} = 3.0$, N is the number of iteration steps, survival time $T_S = 0.02 * N$, $T = 0.02 * N_{max}$ with the maximum number of iterations $N_{max} = 1000$, and ω_k are selected positive weights. The first and second terms determine the accumulated sum of normalised absolute deviations of x_1 and x_2 from zero, similarly for the third term and fourth terms in relation to x_3 and x_4, and the last term when minimised, maximises the survival time.

The essence of the compositional method lies in the method of fitness function evaluation. The choice of evaluation method decides of effectiveness of the compositional method and therefore plays a crucial role. In our approach we decided to define a very simple evaluation method: the fitness function is evaluated for every initial condition and then averaged and assigned to a particular string in controller population.

Fitness function can be modified in order to reward strings which successfully control the system from a large number of initial conditions. One of the simplest methods is to establish threshold values for the objective function and penalize strings that exceed those threshold values (for each init. condition). We set the following threshold values:

$0.3 * avg$, $0.5 * avg$, and $0.8 * avg$ with corresponding penalties as: 500.0, 1000.0, and 2000.0, where avg is a variable representing average fitness of the previous generation:

```
Penalty schedule:
if ObjFun >= 0.3*avg and ObjFun < 0.5*avg
then ObjFun = ObjFun + 500.0;
if ObjFun >= 0.5*avg and ObjFun < 0.8*avg
then ObjFun =  ObjFun + 1000.0
if ObjFun >= 0.8*avg then ObjFun =  ObjFun + 2000.0
```

Please note, that increasing penalty values might 'derail' the evolutionary algorithm. Therefore penalties need to be fine-tuned to focus the EA on selecting strings that perform well for the large number of initial conditions. In some simulations the heavy penalties imposed on fitness function resulted in poor controller performance. To test the impact of above penalty scheme the same simulations were run without any penalties with mixed results that proved that usually not one factor decides on the EA performance but a combination of EA parameters.

The initial population, $P(0) = \{\underset{\sim}{p}_k \ : \ k = 1, \cdots, M\}$, was determined by choosing the a_j as a random integer in $[1, 7]$ where M is the size of the evolutionary population. Full replacement policy was used and for selection process we used tournament selection with size $n_T = 4$.

The new population $P(t + 1)$ is obtained from the old one $P(t)$ by the use of genetic operators such as: selection, crossover, and mutation. Full replacement policy is implemented and requires that the population size remains constant from one generation to the next. A selection process is used to obtain parents for mating in the current generation. We chose tournament selection in which a number of possible parents are selected at random from the population. A tournament is then held to select the two most fit strings and they are used as parents in the next process of crossover to generate children to be passed into the next generation. We used tournament selection with size $n_T = 4$. An elitism strategy is typically used to pass the fittest individuals or copies of the fittest individual to the new population, so that the information encapsulated in the best individual is not lost and the fittest individuals are passed into the next generation.

In the crossover operation a number of 'parent' strings, typically two, are recombined to create 'child' strings. The children are then added to complete the new population. They also undergo mutation by a mutation operator which perturbs or mutates the string structures. With a given probability, usually small, the mutation operator mutates elements of the strings in the population. This ensures satisfactory diversity within the population which is required for the EA to find better approximate solutions to the problem. Mutation was undertaken with probability p_m whose value was determined by a mutation schedule that decreases typically from 0.8 to 0.001 over 1000 generations.

With an appropriate selection of EA parameters and operators, the algorithm is allowed to evolve. The most fit individual is then taken as the best possible solution learnt by this algorithm. Alternatively, the best top ten individuals are amalgamated (averaged) into the control system for the inverted pendulum. The described EA is used to learn fuzzy rules in the HFS that constitutes a control system for the inverted pendulum. A schematic algorithm is given below:

1. Population is randomly initialised: every component of individual string is given a randomly selected value from interval $[1, 7]$.

2. EA parameters are selected: type of inference engine, crossover, mutation schedule, selection method, elitism, fitness function (with a penalty schedule), and number of generations.
3. EA algorithm starts:
   (a) Fitness of the first generation is evaluated.
   (b) Next generation is created using EA operators: selection, crossover, mutation.
   (c) Individual from the population is selected.
   (d) Initial condition is selected from the predefined list.
   (e) Dynamical system is simulated from a given initial condition.
   (f) Final state of state variables and survival time are determined.
   (g) Based on values from 3f temporary fitness function value is evaluated for an individual.
   (h) Penalties are added to the fitness value (if penalty schedule is defined).
   (i) Steps 3d–3h are repeated until all system simulations for every initial condition in the list are performed.
   (j) Average of all temporary fitness values is calculated and assigned to the individual as its fitness.
   (k) Steps 3c–3j are repeated until all individuals in the population have their fitness evaluated.
   (l) Steps 3b–3k repeated until the final generation.
4. Final control system is determined by either selecting the top individual or by averaging top 10 individuals from the final population. Its performance is evaluated by running a simulation of the dynamical system for all initial conditions and counting initial conditions for which the final state variables are within the target region.

5 Simulation results

Methodology described above was implemented in our experiments: 130 simulations were run for different variants of fitness function and different combinations of EA parameters. We illustrate our results on a few selected examples that represent typical results.

The population size was set at 500. Smaller population size is possible but below 300 it is difficult for the EA to maintain the required diversity in population. The EA was terminated at 1000 generations as it was found that the algorithm either finds solution in about $300 - 800$ generations or fails. Changing the weights ω in fitness function had a significant impact on the EA performance. In fact, the fifth component of the fitness function ω_5 (survival time) implicitly contains all other components but by specifying them separately we influence the EA process, i.e., we introduce smaller or bigger bias towards one or another component. Full set of 255 initial conditions was defined for $|x_1| \leq 0.75$, $|x_2| \leq 1.0$, $|x_3| \leq \pi/12$, $|x_4| \leq 1.0$.

By introducing strong elitist strategy we achieved the convergence of the average value of objective function across population close to the minimum value of the objective function. This is desired as an indication of good EA performance resulting in majority of population being valid control systems. In several simulations the average population fitness was on par with the minimal fitness indicating that almost all individuals in the last population represented the control system of the same or very similar quality. In some simulations we amalgamated top ten control strings from the final population into one final controller. The amalgamation resulted in higher percentage of initial conditions from which the controller was able to control the system towards the

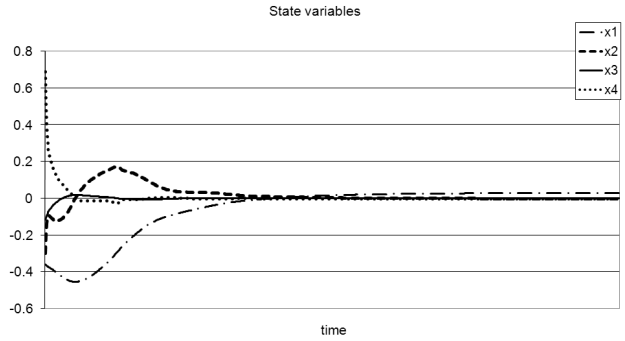

Fig. 3. Typical result: State variables x_k.

target area. The amalgamation of top control strings had adverse effect for the small number of initial conditions for which the controller went quickly out of range of numerical values ('blow-out' effect). With the controller defined as a top string from the final population control was always maintained, i.e, the trajectories converged if not in the target region than relatively close to it. However, as mentioned earlier, the number of initial conditions from which the controller successfully controlled the system to TR dropped. Typically, in most simulations the percentage of initial conditions for which

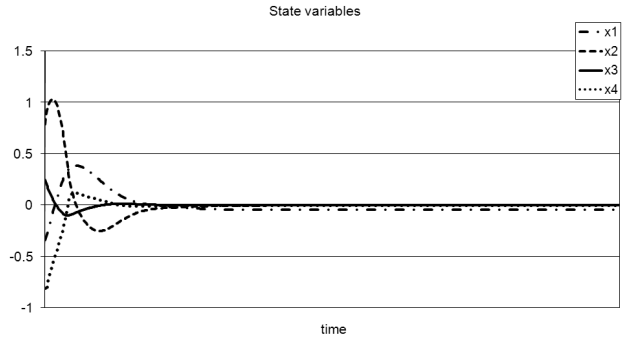

Fig. 4. Good convergence: State variables x_k.

controller successfully controlled the system to TR varied from about 40% to 60% with

a bulk of simulations achieving just about 50% success rate. In many simulations the number of initial conditions from which the controller performance was satisfactory oscillated around 100 (out of 255). This trend might reflect the nature of the inverted pendulum dynamics. We observed that even though the controller often missed the target region the final state variables values were close to TR.

In simulation illustrated in Figure 3 minimum inference engine and penalty schedule were used. Results are shown for initial condition: $\underset{\sim}{x} = (-0.75, -1.0, -0.2618, -1.0)$ and: $\omega_1 = 3000$, $\omega_2 = 100$, $\omega_3 = 100$, $\omega_4 = 0$, $\omega_5 = 2000$. Another typical result is illustrated in Figure 3 for init. condition: $\underset{\sim}{x} = (-0.35, -0.5, -0.1309, 1.0)$ and: $\omega_1 = 1000$, $\omega_2 = 0$, $\omega_3 = 1000$, $\omega_4 = 0$, $\omega_5 = 3000$. In simulation illustrated in Figure 4 we used product inference engine and no penalty schedule. Results shown are for initial condition: $\underset{\sim}{x} = (-0.75, -1.0, -0.2618, -1.0)$ and: $\omega_1 = 3000$, $\omega_2 = 100$, $\omega_3 = 100$, $\omega_4 = 0$, $\omega_5 = 2000$. In this particular simulation a very good state variables convergence was achieved but only for 99 out of 255 initial conditions. All state variables converged very quickly to zero, except x_1 with values remaining about 0.04 from the origin.

6 Conclusions

In this paper we have examined the hierarchical fuzzy control of the simple inverted pendulum and used evolutionary algorithms to learn a fuzzy controller for a single hierarchical topology over the whole set of initial conditions. We defined the set of 255 initial conditions in state space that is viable in terms of inverted pendulum dynamics. We excluded initial conditions that were in our opinion too extreme and would make control process virtually impossible. This fact reflects physical reality of inverted pendulum dynamics; if the system starts from completely different initial conditions it is unlikely to find relatively small fuzzy rule base capable of handling every possible dynamics of the system. For the set of this size and range, the task of controlling the inverted pendulum from every initial condition in the set proved unrealistic. We set our goal at achieving control to the target area for as many initial conditions as possible.

We designed the compositional method for the inverted pendulum system and proved that with the right combination of EA parameters the resulting fuzzy control system is capable to control the system from the wide range of initial conditions. Our results show that with improvements in the EA parameters (especially fitness function definition) better results are possible.

Compositional method provides a control system (not just a method for designing a controller) that once developed is capable of controlling the dynamical system from a wide range of initial conditions. In many practical applications this is a significant advantage as computational time for designing controller for every initial condition separately may render a method unfeasible.

References

1. Castillo O., Cazarez N., Rico D.: Intelligent Control of Dynamic Systems Using Type-2 Fuzzy Logic And Stability Issues. In: International Mathematical Forum. Vol.1, No. 28, pp. 1371–1382, (2006)

2. Zajaczkowski, J., and Stonier, R.J.: Co-evolutionary algorithm for hierarchical fuzzy control fof the inverted pendulum. In: Proceedings of WCCI2006, IEEE International Conference on Fuzzy Systems (CD-ROM). Vancouver, Canada, (2006)
3. Beceriklia Y., Koray Celik B.: Mathematical and Computer Modelling. In: Proceedings of the International Conference on Computational Methods in Sciences and Engineering 2004. Vol. 46, Issues 1-2, pp. 24–37, (2007)
4. Stonier R.J., Zajaczkowski J.: Hierachical fuzzy controllers for the inverted pendulum. In: Proceedings of CIRAS 2003. Singapore (CD-ROM), (2003)
5. Qiao F., Zhu Q.M., Winfield A., Melhuish C.: Fuzzy sliding mode control for discrete nonlinear systems. In: Transactions of China Automation Society. Vol. 22, No. 2, pp. 313–315, (2003)
6. Suykens J.A.K., Vandewalle J., de Moor B.: Optimal control by least squares support vector machines. In: Neural Networks 14, pp. 23–35, (2001)
7. Yi J., Yubazaki N.: Stabilization fuzzy control of inverted pendulum systems. In: Artificial Intelligence in Engineering. Vol. 14, Issue 2, pp. 153–163, (2000)
8. Shuliang L., Langari R.: Hierarchical fuzzy logic control of a double inverted pendulum. In: Fuzzy System 2000, FUZZ IEEE 2000, The Ninth IEEE International Conference. Vol. 2, pp. 1074–1077, (2000)
9. Huang S.J, Huang C.L: Control of an Inverted Pendulum Using Grey Prediction Model. In: IEEE Transactiona On Industry Applications. Vol. 36, No. 2, (2000)
10. Stonier R.J., Stacey A.J., Messom C.: Learning fuzzy controls for the inverted pendulum. In: Proceedings of ISCA 7th International Conference on Intelligent Systems. pp. 64–67, Melun, Paris, (1998)
11. Lin C.T., Lee C.S.G.: Neural Fuzzy Systems. Prentice Hall, (1996)
12. Wang L.X.: A Course in Fuzzy Systems and Control. Prentice-Hall, (1997)
13. Zajaczkowski J., Stonier R.J.: Analysis of hierarchical control for the inverted pendulum. In: Proceedings of Complex2004. Cairns (CD-ROM), pp. 350–374,(2004)
14. Michalewicz Z.: Genetic Algorithms + Data Structures = Evolution Programs. 2nd Ed., Springer Verlag, (1994)