SCIS & ISIS 2006

Paper ID: 100005

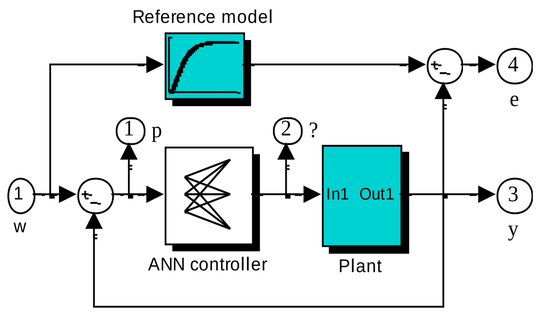
Design of Neural Network Controller for Robotic Manipulator Based on Evolutionary Algorithms

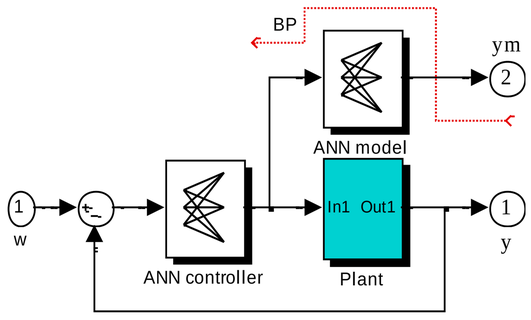
Avdagic Zikrija, Konjicija Samim and Lacevic Bakir

University of Sarajevo, Faculty of Electrical Engineering

In this work, capabilities of a feed-forward neural network regarding control of the complex object are investigated. Neural controllers have been trained by evolutionary strategy and genetic algorithm with adaptive mutation and crossover probabilities. A specific model of an aggressive selection operator is proposed along with one way of co-evolution of the crossover and mutation rates. Also, different mechanisms of operator adaptation were compared in the sense of the resulting controller performance. Finally, the measurement results, taken from the object (hydraulically driven two-joint robot arm) are presented.

KNOWN AND UNKNOWN SIGNALS



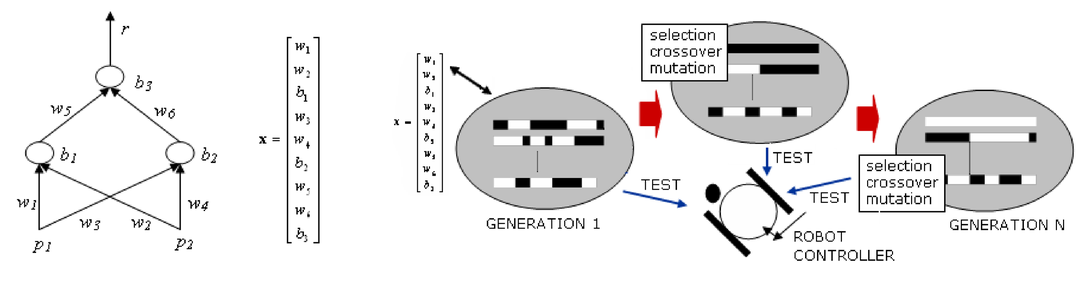


- Reference model
- Sistem output
- Unknown controller output

- ANN direct model
- BP algorithm

Another optimization algorithm?

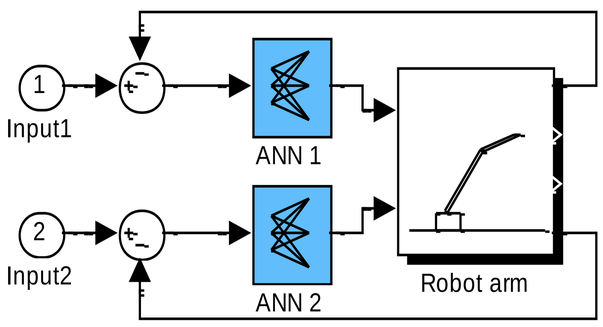
EVOLUTIONARY APPROACH

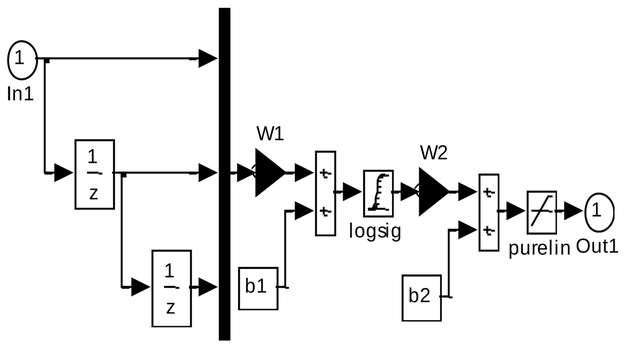


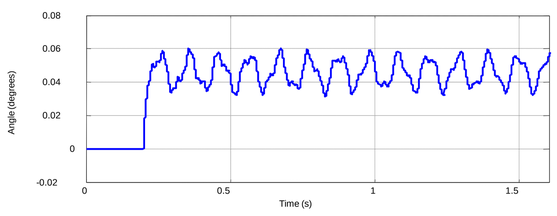
Basic evolotionary robotics methodology

- Encoding ANN parameters into chromosome
- One controller tested at a time
- Survival of the fittest controllers

CONTROL SYSTEM TOPOLOGY



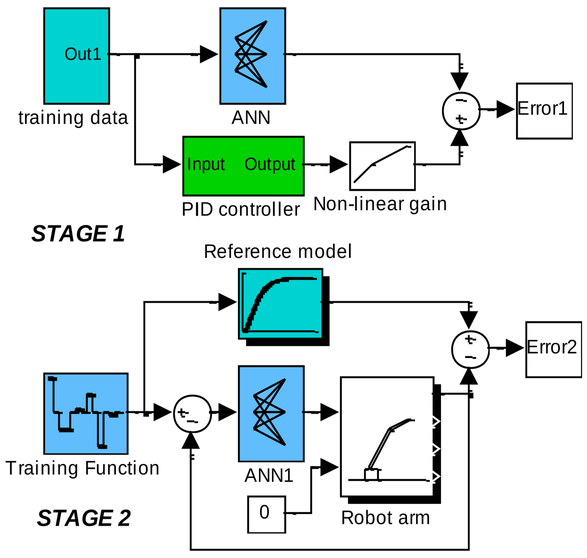




Impulse response of the first link angle

- Two-joint robot arm
- Decentralized control
- Two independent ANN controllers
- One hidden sigmoidal layer (10 neurons)

EVOLUTION OF THE CONTROLLERS



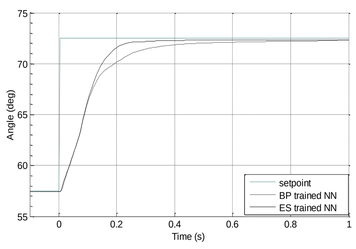
$$F = \sum_i \ln(1 + |e_i|)$$

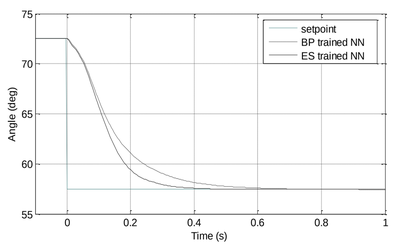
Objective function

- Two steps evolution
- Cloning the optimal PID
- Training ANN in a closed loop

Evolution has been done in two steps. First step consists of cloning the optimal PID controller in order to obtain a qualitative population for further evolution. Second step is training the ANN controller in a closed loop with the model of the object. In the first step, a nonlinear output gain is used for compensating the influence of gravity. When the cloning of the PID controller is done, the resulting population is used as a starting population for second stage of evolution. Figure on the left shows the training schema for the ANN controller of the first link angle. The angle of the second link is kept constant. Training the ANN controller of the second link angle is completely analogous.

EVOLUTIONARY STRATEGY APPLICATION

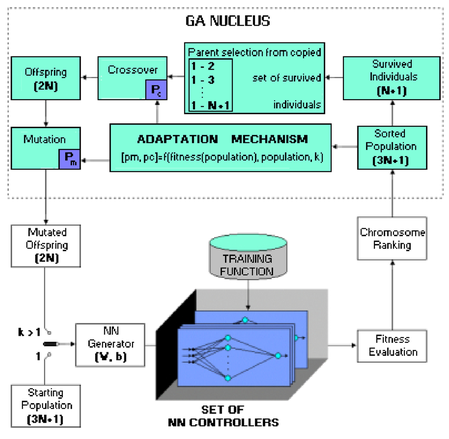


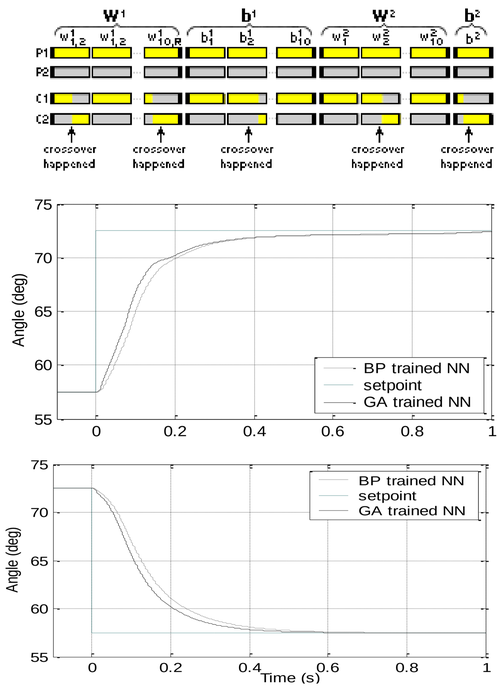


	BP	ES
Rise time t_r (sec)	0.2520	0.1770 (42% better)
Fall time t_p (sec)	0.3020	0.2150 (40% better)
Steady state error (°)	0.0630	0.0380 (66% better)

Inputs to the trained controller were error , its integral and derivation. ES(1,4) was implemented, with univariate mutation operator, and without recombination operator. Standard deviation of mutation was adjusted in dependence on the effect of previously applied mutation, so that simultaneous successful mutations increased the deviation widening in that way the search area, whereas simultaneous unsuccessful mutations decreased the deviation by certain amount narrowing the search area.

GENETIC ALGORITHM APPLICATION





Out of the starting population (3*N*+1 chromosomes, where *N* is a positive integer), the ANN generator calculates weights and biases for each controller (switch position is "1"), whose fitness is evaluated, after simulation has been done. After that, the sorting of individuals is being done in such a way that the first individual is the best one. The selection operator rejects the last 2*N* individuals while the copy of survived *N*+1 chromosomes is being made. The set of copied survivors forms the mating pool. Then the best chromosome from the above mentioned set takes part in crossover with each of the remaining *N* chromosomes from the mating pool. The result is the offspring formed from 2*N* members, which are then left to the mutation operator. Out of the mutated offspring, NN generator forms 2*N* controllers (meanwhile, the switch position has changed to "k" and remains unchanged) whose fitness is evaluated in a mentioned way. Now, the mutated offspring with its fitness joins the set of original parents (of whom the copy has been made) and the population is completed again (3*N*+1 individuals). The next step is sorting this population and the process is being repeated until the termination criterion is fulfilled (maximal number of generations). It is obvious that the selection pressure of this algorithm is extremely great, but it is expected that the adaptation mechanism should preserve the variety of population in order to prevent the premature convergence. Recombination is being done in a way that every weight of the first parent crosses over with corresponding weight of the second parent (corresponding weights have identical indices, such as locations). Results of these "elementary" one-point crossovers are new weights that are then joined in two complete chromosomes representing the final product of the crossover operation (Fig. 8b). It is obvious that the probability of elementary crossover controls the expected number of crossover points of the whole chromosome. In every generation, the adaptation mechanism determines crossover and mutation probabilities, which, in general, depend on the state of population and on the generation number.

	BP	GA
Rise time t_r (sec)	0.2689	0.2532 (6.2% better)
Fall time t_p (sec)	0.2970	0.2583 (15% better)
Steady state error – positive step (°)	0.0265	0.0258 (2.7% better)
Steady state error - negative step (°)	0.0292	0.0305 (4.3% worse)

ADAPTATION MECHANISMS

Pre-deterministic

1)

$$p_m(t) = p_{m\min} + (p_{m\max} - p_{m\min})\frac{t_{\max} - t}{t_{\max}}$$

$$p_c = const$$

feedback-deterministic

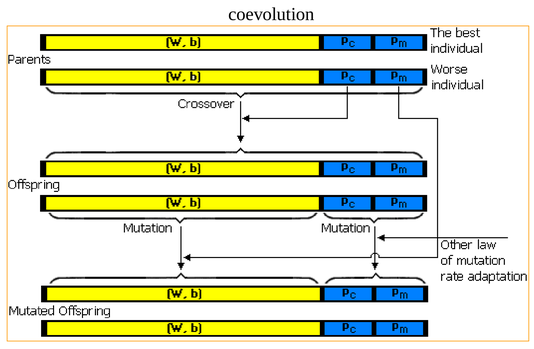
2)

$$p_c = \begin{cases} p_{c\max} - (p_{c\max} - p_{c\min})e^{-\frac{dp\_mid Fit - \min Fit}{K_1 \max Fit - \min Fit}}, & \max Fit \neq \min Fit \\ p_{c\min}, & \max Fit = \min Fit \end{cases}$$

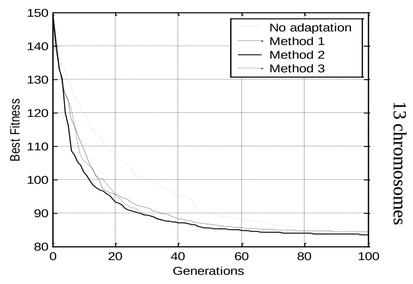
$$p_m = \begin{cases} p_{m\min} + (p_{m\max} - p_{m\min})e^{-K_2 \cdot \frac{Fit - \min Fit}{\max Fit - \min Fit}}, & \max Fit \neq \min Fit \\ p_{m\max}, & \max Fit = \min Fit \end{cases}$$

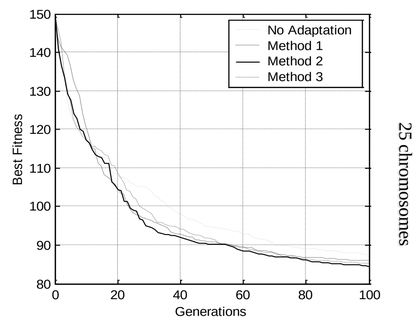
coevolution

3)



Method	25 chromosomes		13 chromosomes	
	AF	SDF	AF	SDF
No Adaptation	84.4206	0.2829	87.5134	1.1934
1	84.1212	0.5626	85.8405	0.7106
2	83.5713	0.1390	84.5193	0.7465
3	83.2636	0.6074	85.1524	0.9759





CONCLUDING REMARKS

- EA Improved controller performance
- No restrictions when choosing objective function
- No need to know the controller outputs
- Operators adaptation ensured better EA performance, especially for smaller populations
- EA outperformed BP