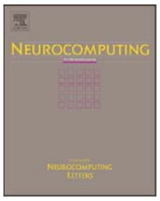
Neurocomputing 72 (2009) 2806–2814

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



A method to learn the inverse kinematics of multi-link robots by evolving neuro-controllers ☆

José Antonio Martín H. [a,*], Javier de Lope [b], Matilde Santos [c]

[a] *Dep. de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Spain*
[b] *Department of Applied Intelligent Systems, Universidad Politécnica de Madrid, Spain*
[c] *Dep. de Arquitectura de Computadores y Automática, Universidad Complutense de Madrid, Spain*

ARTICLE INFO

Available online 16 April 2009

Keywords:
Inverse kinematics
Neuro-evolution
Neuro-controllers
Multi-link robots

ABSTRACT

A general method to learn the inverse kinematic of multi-link robots by means of neuro-controllers is presented. We can find analytical solutions for the most used and well-known robots in the literature. However, these solutions are specific to a particular robot configuration and are not generally applicable to other robot morphologies. The proposed method is general in the sense that it is independent of the robot morphology. The method is based on the evolutionary computation paradigm and works obtaining incrementally better neuro-controllers. Furthermore, the proposed method solves some specific issues in robotic neuro-controller learning: it avoids any neural network learning algorithm which relies on the classical supervised input-target learning scheme and hence it lets to obtain neuro-controllers without providing targets. It can converge beyond local optimal solutions, which is one of the main drawbacks of some neural network training algorithms based on gradient descent when applied to highly redundant robot morphologies. Furthermore, using learning algorithms such as the neuro-evolution of augmenting topologies it is also possible to learn the neural network topology which is a common source of empirical testing in neuro-controllers design. Finally, experimental results are provided when applying the method to two multi-link robot learning tasks and a comparison between structural and parametric evolutionary strategies on neuro-controllers is shown.

1. Introduction

One of the most traditional areas in robotics is dedicated to the study of kinematics of articulated robots. As it is well known, robot kinematics establishes a mechanism to determine the relationship between the joint and Cartesian coordinates. The direct kinematics problem has been successfully solved in the middle of the past century with the pioneer work of Denavit and Hartenberg [6]. Currently there exist automatic procedures to assign the different parameters to every robot's joint and obtaining the end-effector position and orientation referred to a base frame for each point expressed in the joint coordinate frame.

The inverse kinematics problem consist in determining a transformation between the external reference frame, generally expressed in terms of the true goal coordinates, and the internal reference frame of the robot, generally expressed in articulation states. Unfortunately the inverse kinematics problem has not been solved analytically in a general way. We can find analytical solutions for the most used and well-known robots in the literature. However, the main drawback of these solutions is that they are specific to a particular robot configuration and are not applicable to other robots. The problem gets worst if we consider that currently several kinds of new articulated robots are being proposed for different tasks. Humanoid and modular robots are the usual topics in the most important international conferences on robotics. These multi-articulated robots make inapplicable some of the classical procedures to calculate the inverse kinematics mainly due to such methods assume some specific configurations which are not used in the new articulated robots. These new articulated robots are inherently multi-redundant. For example, a simple biped robot needs 12 degrees-of-freedom (DOF) to reach the most common configurations required for realistic postures. A whole humanoid robot has around 30 DOFs, depending on the number of modeled joints. There is no predefined value for the case of modular and serpentine robots. This degree of redundancy makes the development of an analytic solution for the inverse kinematics practically unfeasible.

Several methods have been proposed to solve the inverse kinematics of multi-link robots by means of artificial neural

☆ *Software*: All the software packages, simulators and the experimental programs used in this research are publicly available through the web site: http://www.dia.fi.upm.es/~jamartin/download.htm.
* Corresponding author. Tel.: +34 91 394 7646.
E-mail addresses: jamartinh@fdi.ucm.es (José Antonio Martín H), javier.delope@upm.es (J. de Lope), msantos@dacya.ucm.es (M. Santos).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.08.021

networks (ANNs). Some approaches [3,9,11] suggest following specific procedures for the sample data generation or for the training process. Martinetz et al. [11] proposed a method to be applied to manipulators with no redundant DOF or with a redundancy resolved at training time. Jordan and Rumelhart [9] suggested first training a network to model the forward kinematics and then prepending another network to learn the identity. Other alternatives [10] include the use of particular networks like Hopfield neural networks, modular neural networks [1] and recurrent neural networks [16]. Multi-layer perceptrons [3,12,13] are also used to learn the inverse kinematics of redundant manipulators and biped robots. de Lope et al. [3–5,13] have previously proposed several approaches for solving the inverse kinematics of biped robots based on the classical (input-target) training methods of ANNs. The method uses a set of sample pairs which relate the position and orientation of a foot to its joint coordinates. The training set is generated according to some specific criteria which avoids unstable or unnatural configurations and it is also restricted to particular gaits, for instance all the positions used for single steps, for lateral displacements, etc. So it can be generalized to a great variety of different movements.

In this paper we propose a general method in the sense that it is independent of the robot morphology. We compare two different learning algorithms in order to obtain the best neuro-controllers to solve the inverse kinematics of two models of robot manipulators. Thus the main goal of this work is to obtain a general method to learn the inverse kinematics of multi-link robots.

This paper is organized as follows. In Section 2 the proposed method is described in detail, explaining the main goals of our approach and presenting an evolutionary framework for the adaptation of neuro-controllers for multi-link robots, next in Section 3 a concise discussion about evolutive methods and neuro-evolution is presented, in Section 4 the experimental framework is defined introducing two multi-link robot models and a description of the proposed experiments is also presented. In Section 5 experimental results are provided when applying the proposed method in two learning tasks and a comparison between structural and parametric evolutionary strategies on neuro-controllers is shown. Finally, in Section 6 we present the conclusions and some further lines of research.

2. Description of the proposed method

Here we propose a method based on the adaptation of neural networks by means of evolutionary computation. The proposed method is developed in order to achieve three main goals:

(i) To avoid any neural network learning algorithm which relies on the classical supervised input-target learning scheme. Hence it would allow to obtain neuro-controllers without the requirement of providing targets or correct answers, which in this case are unknown in prior, or large amounts of engineering specifications on the particular model of the robot to be used.
(ii) To converge avoiding local optimal solutions which is one of the main drawbacks of some neural network training algorithms based on gradient descent when applied to highly redundant robot morphologies.
(iii) Learning the neural network topology "on-the-fly" which is a common source of empirical testing in optimal neuro-controllers design.

In order to prove how these goals are achieved, we apply the method to two multi-link robot learning tasks and a comparison between structural and parametric evolutionary strategies on neuro-controllers is shown.

2.1. *An evolutionary framework for the adaptation of neuro-controllers*

In order to learn the inverse kinematics of a robot we define an artificial neural network whose input represents the desired robot configuration in the natural reference frame, that is, in the reference frame in which the goals of the problem are expressed. Thus the response of the network will be a vector that contains the final state of the robot articulations in such a way that, by following its direct kinematics, the state of the robot is updated and the final position is reached.

The way of evolving a neuro-controller consists in defining a vector of real numbers that represent the synaptic weights of the network. In this way, the vector will contain as many components as synaptic weights has the network. Then for the obtention of a phenotype (a neuro-controller) we proceed to extract the synaptic weights vector and to form the corresponding matrices for the weights of the hidden and output layers. Finally, by means of two simple matrix multiplications (dot product) we obtain the response of the neuro-controller.

The operation of a multi-layer perceptron with one single hidden layer and with the hyperbolic tangent (tan h) as the activation function can be described as follows:

Given an input vector (inputs):

$$a_i = [\text{inputs}, 1],$$

where 1 stands for the bias input and a_i for the activation of the input layer. We proceed to calculate the network response by following a simple rule:

$$a_o = \tanh(\tanh(a_i \cdot w_i) \cdot w_o), \tag{1}$$

where the matrices w_i and w_o are, respectively, the synaptic weights of the hidden and output layers as shown in Fig. 1.

This formulation can be generalized to incorporate multiple hidden layers and any other activation function f by following a simple chain formulation:

$$a_o = f_n \cdots (f_3(f_2(f_1(a_i \cdot w_1) \cdot w_2) \cdot w_3) \cdot \cdots w_n), \tag{2}$$

where each f_i is an arbitrary activation function, the matrices w_i represent the synaptic weights of the different layers of artificial neurons and a_o is the activation of the output layer.

Thus, we can obtain a measurement of the performance evaluation of a neuro-controller by measuring the difference between the final state of the robot and the desired state (inputs vector) as shown in Fig. 2.

The evaluation cycle of the neuro-controller can be described by the next steps:

(i) Set the input I of the neuro-controller to the desired final state of the robot in the natural reference frame where the goals are defined.
(ii) Obtain the neuro-controller response as the vector O where the final state is expressed in the robot reference frame (joint and articulation space).
(iii) Apply the control action, that is, apply the vector O to a direct kinematics module in order to obtain the final state F of the robot.
(iv) Proceed to evaluate the quality of the neuro-controller using a measure of the difference between the desired position I and the current robot position F.

This procedure must be repeated for a significative number of random desired samples in order to assure that for a certain error

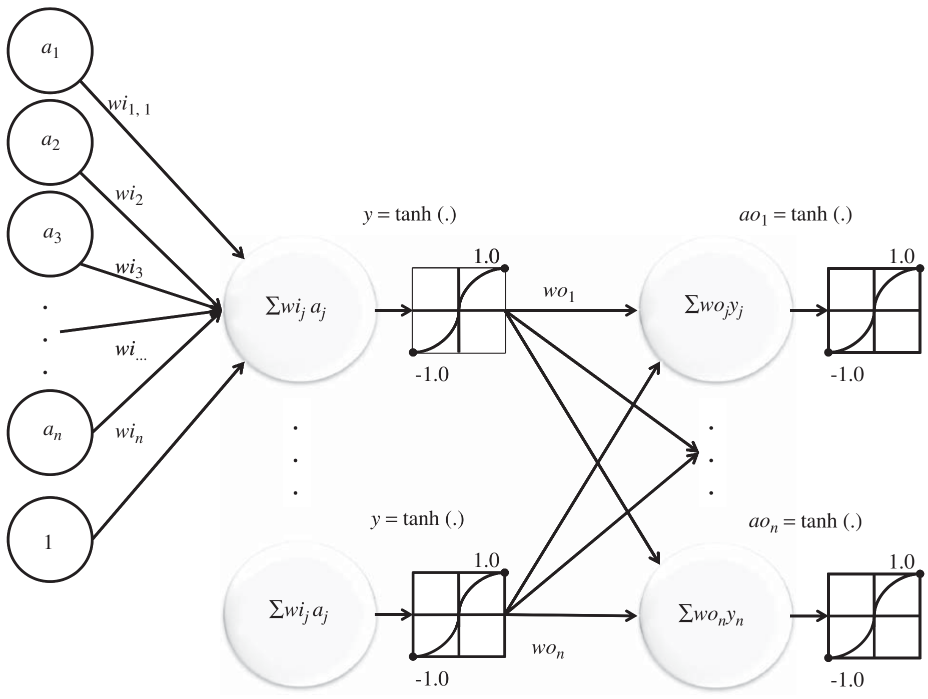


Fig. 1. The operation of a multi-layer perceptron with one single hidden layer and with the hyperbolic tangent (tanh) as the activation function.

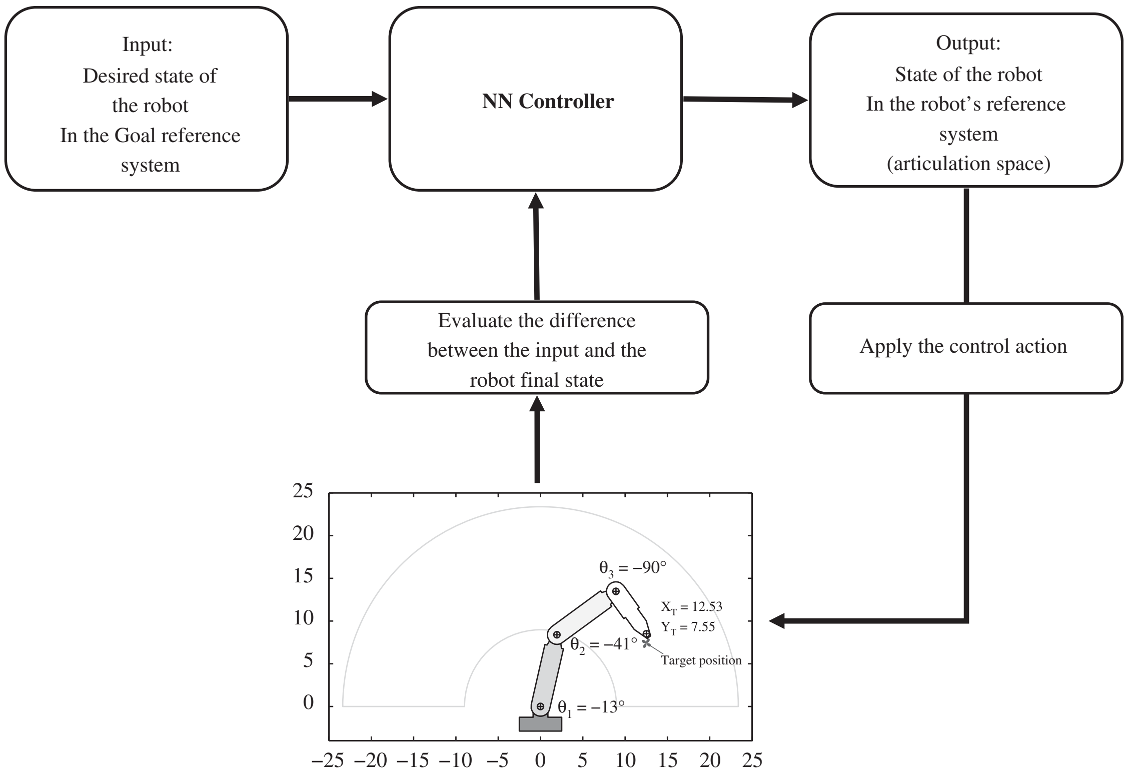

Input:
Desired state of the robot In the Goal reference system
NN Controller
Output:
State of the robot In the robot's reference system (articulation space)
Evaluate the difference between the input and the robot final state
Apply the control action
$\theta_3 = -90°$
$\theta_2 = -41°$
$\theta_1 = -13°$
$X_T = 12.53$
$Y_T = 7.55$
Target position

Fig. 2. Evaluation cycle of an evolved neuro-controller.

margin the controller has learned and it generalizes well enough over the complete workspace of the robot.

3. Evolutive methods and neuro-evolution

There are two major methods to train artificial neural networks by means of evolutionary strategies. The first one varies the network weights trying to minimize the error in the output layer related to a previously defined dataset or a simulated environment. Apart from this *parametric* evolution, there is also a kind of *structural* learning in which the network topology is also modified [2]. Again, two categories can be defined: the *constructive* methods, in which starting from a simple network the nodes and links are added, and *destructive* methods in which the process starts with large networks and the superfluous components are pruned off. We introduce and compare two methods for neural network training, the first is an example of the parametric approach and the second modifies the structural aspects of the neural networks. Here we propose the use of two “state-of-the-art” evolutionary techniques:

(i) The covariance matrix adaptation evolution strategy (CMA-ES).
(ii) Neuro-evolution of augmenting topologies (NEAT).

The covariance matrix adaptation evolution strategy [7] is one of the most powerful parametric evolutionary algorithms for real-valued optimization [8]. By this reason it has been considered in this paper to be compared to the structural neuro-evolution of augmenting topologies [15] algorithm which is also a very well-known neuro-evolutionary algorithm.

CMA-ES [7] is a de-randomized method to adapt the covariance matrix of the multivariate normal mutation distribution in the evolutionary strategy. The covariance matrix describes the pairwise dependencies between the variables. The adaptation of the covariance matrix leads to learning a second order model of the underlying objective function, similar to the approximation of the inverse Hessian matrix in the quasi-Newton method in classical optimization. In general the adaptation is based on the idea of increasing the probability of a successful mutation step. The covariance matrix is changed so that the probabilities of the successful steps of the last generation are increased. The adaptation of the multivariate normal distribution in the CMA evolution strategy consists of two parts: (1) adaptation of the covariance matrix, which can be interpreted as executing an online-PCA and (2) adaptation of the overall step length (step-size), implemented by measuring the length of the step between generations of the population.

NEAT [15] is a method for training artificial neural networks that is proposed as an alternative to other methods which does not modify the network topology during the learning process. Both kinds of methods present advantages for a particular problem. The advantages are related to save time when avoiding to find the right number of hidden neurons by methods that modify topologies. However, these methods may make the search more difficult.

The NEAT’s genomes are linear representations of a network connectivity. Thus, the genes are easily lined up during the crossover. Each genome includes a list of connection between genes, each of which refers to two nodes connected. Each connection between gene specifies the in-node, the out-node, the weight, a value which determines if the connection is enabled, and a value, the *innovation number*, that represents the instant in which the gene was added to the genome and was used during the crossover to create more easily an offspring. By using the innovation number mechanism, it can be easily determined which genes match up with which. The basic idea is to use a counter that is incremented when a new gene appears, assigning the value at that moment to the gene. So, individuals which share in genetic information can be detected easily due to the innovation number in their genes. Then, the crossover generates a new offspring from two parents in which shared genetic material is included and also the new contributed genes which are only present in one of the parents. The mutation operator can change both the connection weights and the network structure. The weights are mutated as usual, each connection may or may not be perturbed at each generation. The network structure mutation expands the size of the genome by adding one or more genes and enabling or disabling node connections.

4. Experimental application framework definition

Experiments have been carried out on two multi-link robot models: a three-link-planar robot and a SCARA robot. The objective of each experiment is that the robots learn their inverse kinematics in order to reach a desired configuration in the joint and Cartesian spaces.

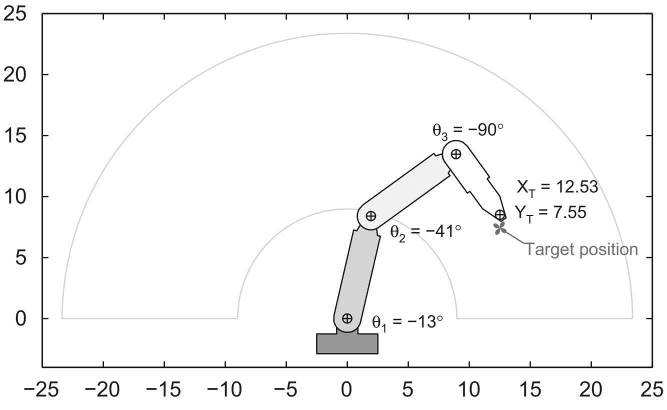


Fig. 3. Three-link-planar robot model.

Table 1
Three-link-planar robot D–H parameters.

i	θ_i	d_i	a_i	α_i
1	θ_1	0	$a_1 = 8.625$	0
2	θ_2	0	$a_2 = 8.625$	0
3	θ_3	0	$a_3 = 6.125$	0

4.1. The three-link-planar robot

The three-link-planar robot includes some redundancy in the calculus of the inverse kinematics by means of analytic procedures. For this reason, the verification of our methods on this kind of mechanical structures is very interesting. The same approach could be applied for controlling the movement and body coordination of, for instance, snake-like robots. Fig. 3 shows a diagram of this robot.[1] Its Denavit–Hartenberg [6] (D–H) parameters are shown in Table 1 and its direct kinematics is defined in (3). Both the D–H parameters and the direct kinematics matrix defined in (3) are properties that uniquely define an articulated mechanical device, i.e. an articulated robot (manipulator) in standard terms (see [14] for a detailed explanation)

$$T^3 = \begin{pmatrix} C_{123} & -S_{123} & 0 & a_1C_1 + a_2C_{12} + a_3C_{123} \\ S_{123} & C_{123} & 0 & a_1S_1 + a_2S_{12} + a_3S_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3}$$

where S_{123} and C_{123} correspond to $\sin(\theta_1+\theta_2+\theta_3)$ and $\cos(\theta_1+\theta_2+\theta_3)$, respectively (and the corresponding notation stands for S_1, S_{12}, C_1 and C_{12}), θ_1, θ_2 and θ_3 define the robot joint coordinates—θ_1 for the first joint, located near the robot base, θ_2 for the middle joint and θ_3 for the joint situated in the end extreme, and a_1, a_2 and a_3 correspond to the physical length of every link, first, second and third, respectively, numbered from the robot base.

[1] The original model is by Matt Kontz from Walla Walla College, Edward F. Cross School of Engineering, February 2001.

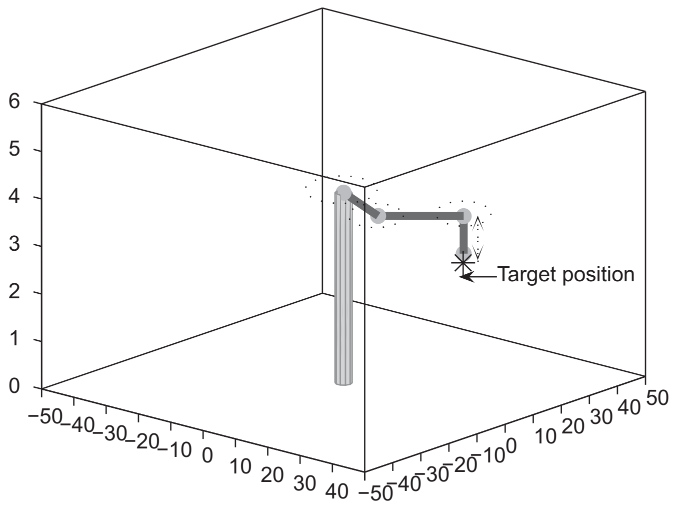


Fig. 4. SCARA robot model.

Table 2
SCARA robot model D–H parameters.

i	θ_i	d_i	a_i	α_i
1	θ_1	0	$a_1 = 20$	0
2	θ_2	0	$a_2 = 20$	π
3	θ_3	0	0	0
4	0	d_4	0	0

4.2. The SCARA robot

The SCARA robot was selected since it is widely used in the industry. It has well-known properties for its use in manufacturing cells assisting conveyor belts, electronic equipment composition, welding tasks, and so on. In this case we are using a real three-dimension robot manipulator. A physical model of the SCARA robot is shown in Fig. 4 with its corresponding D–H parameters presented in Table 2. The direct kinematic matrix is formulated in (4)

$$T^4 = \begin{pmatrix} C_{12-3} & -S_{12-3} & 0 & a_1C_1 + a_2C_{12} \\ S_{12-3} & C_{12-3} & 0 & a_1S_1 + a_2S_{12} \\ 0 & 0 & -1 & -d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4}$$

where S_{12-3} and C_{12-3} correspond to $\sin(\theta_1 + \theta_2 - \theta_3)$ and $\cos(\theta_1 + \theta_2 - \theta_3)$, respectively (and the corresponding notation for S_1, S_{12}, C_1 and C_{12}), $\theta_1, \theta_2, \theta_3$ and d_4 are the joint parameters for the shoulder, elbow, wrist and prismatic joints, respectively, and a_1 and a_2 are the lengths of the arm links.

4.3. Experiments design

The control task consists in reaching a given target position (goal position) of the robot end-effector. These goal positions are defined in such a way that they are always achievable for the robot. The robots are controlled by means of assigning a desired angle position to each articulated link and following thus its direct kinematics. Therefore, if the goal (target position) is to reach the point p, then a controller program must provide the robot with a vector $\widehat{\Theta}(p)$ of angular values in order that when applying a direct kinematics method using $\widehat{\Theta}(p)$, the final position (p') of the robot end-effector minimizes the MSE error $\|p - p'\|^2$ as shown previously in Fig. 2.

Thus, by means of different trials (repetitive evaluation cycles), a model of the robot inverse kinematic should be learned by the system using an evolutionary program that will evolve a neuro-controller. For learning purposes we used a fixed uniform random sample of 70 different target positions (a training set). Although using a non-fixed random set of targets may force the learning process to obtain initial networks that generalize well over the complete space of targets, the inclusion of non-fixed training targets add a huge amount of complexity to the optimization cost function due to the effect of evaluating individuals for different target positions, which is actually noise in the fitness function, the evolutionary process may become unstable and can result in poor performance and undesirable side effects. For this reason we prefer a fixed set of training target points in this kind of learning problems combined with an adequate topology design in order to prevent over-fitting and thus promoting good generalization capabilities. Finally, generalization tests should be carried out with a set of new random target positions to assure a proper learning.

For the evolutionary methods scheme we have selected for all of our experiments a population size of 50 individuals and a total of 1000 generations per learning trial. These values, although can be improved in order to gain better results for a particular method, are selected as the minimal feasible requirements for an evolutionary algorithm to work properly in the selected tasks.

The experiments can be divided in two categories depending on the selected evolutionary approach:

(i) A fixed topology multi-layer perceptron evolved with the CMA-ES algorithm.
(ii) Parametric and topology learning through the NEAT algorithm which incrementally learn the network weights and at the same time learn the more appropriate network topology.

Since CMA-ES is applied to a fixed topology network the process of designing the individual's genotype (vector of real numbers) and phenotype (neural networks) is trivial. Each individual's genotype is a vector whose length is determined by three fixed variables:

(i) The number of input nodes $(ni + bias)$.
(ii) The number of nodes in the hidden layer (nh).
(iii) The number of output nodes of the multi-layer perceptron (no).

Thus the genotype will be formed by all the weights of each layer, say

(i) The weights of the input layer:

$$|w_i| = (ni + 1) \times nh.$$

(ii) The weights of the output layer:

$$|w_o| = nh \times no.$$

Hence the genotype will be a vector of real numbers of cardinality:

$$N = |w_i| + |w_o|.$$

Thus, for constructing an individual's phenotype, that is, its respective neural network, we need just to extract the corresponding segments of the genotype vector in order to create the weights matrix of each layer. Then we can test the behavior of any evolved ANN with the next simple procedure.

```
function NNResponse ( inputs, individual )
  $w_i \leftarrow$ individual$[1 \cdots (ni+1) \times nh]$
  $w_o \leftarrow$ individual$[((ni+1) \times nh + 1) \cdots \mathrm{N}]$
  $a_i \leftarrow [inputs; 1]$—1 for bias node
  $a_h \leftarrow \tanh(a_i \cdot w_i)$
  $a_o \leftarrow \tanh(a_h \cdot w_o)$
return $a_o$
```

For the NEAT algorithm the process of transforming the individual's genotype involves the determination of the number of hidden nodes and the respective non-disabled connections. Basically the process consists in a loop: any starting node has to find its corresponding end nodes in order to contribute to the total activation of these end nodes. This process is repeated until every connection has been processed. For NEAT networks we use the same activation function as in the CMA-ES networks (hyperbolic tangent) which allows to handle in an easier way both positive and negative values. Also we have determined empirically that this activation function (tan h) is best suited for this kind of task where the interesting output values are continuous instead of on–off (0,1).

We have used several functional parameters for the NEAT algorithm; only the more relevant parameters which are also the most problem dependent are presented here.

- connected input nodes = all, which means that the algorithm will start with all enabled inputs node connections.
- mutation probability of adding a node = 0.002.
- mutation probability of adding a connection = 0.002.
- mutation probability of mutating weight = 0.5.
- mutation weight cap = 1000. Weights will be restricted from (−mutation weight cap to + mutation weight cap).
- mutation weight range = 1; random distribution with width mutation weight range, centered on 0.
- mutation probability of a gene being re-enabled = 0.25. Probability of a connection gene being re-enabled if it was inherited disabled.

As we can see the "probability of adding a node" and "probability of adding a connection" parameters have been fixed to a very small value. This has the purpose of preventing premature growth of the network topology letting small topologies to have enough time to converge.

The main evaluation procedure for both kinds of methods is shown.

```
function EvaluateIndividual ( individual )
  for each goal $i \in [1,70]$
    $p \leftarrow$ goal$(i)$
    action $\leftarrow$ NNResponse ( $p$, individual )
    $p' \leftarrow$ Robot.DoAction ( action )
    $MSE = MSE + \|p - p'\|^2$
return MSE
```

5. Experimental results

Table 3 shows the complete set of results for every experiment for both robots and methods. In Table 3 the results are ordered from lower to upper error values and also, in the bottom rows, the mean (μ) and the standard deviation (σ) are shown.

Table 4 shows the results of a t-test with unequal variance for all the experiments and Table 5 shows the same tests but for a selection of the best 10 runs for every robot and method.

Table 3
Results obtained for the planar and SCARA robots, the minimum mean squared error (MSE) reached in a learning experiment; the mean (μ) and the standard deviation (σ).

	Planar robot		SCARA robot	
	CMA-ES	NEAT	CMA-ES	NEAT
MSE	0.1976	4.1564	0.0529	10.3440
	0.2534	4.4018	0.1127	11.7130
	0.4418	4.4085	0.1185	12.7866
	0.4564	4.4321	0.1378	12.9188
	0.5899	4.5470	0.1478	12.9759
	0.6536	4.6371	0.1626	13.0571
	0.7082	4.7554	0.1626	13.1727
	0.7083	4.7597	0.1731	13.2791
	0.7170	4.8271	0.1753	13.9484
	0.7344	4.8803	0.1993	13.9895
	0.7430	4.8807	0.2168	14.1789
	0.8494	4.8845	0.2803	14.5171
	0.8829	4.8948	0.3026	14.6569
	1.4925	4.9101	0.3035	14.7743
	1.6220	4.9245	0.3305	15.1011
	1.7418	4.9577	0.3980	15.2932
	3.9612	4.9630	0.5338	15.9752
	4.4558	4.9940	0.5338	16.0734
	4.8925	5.0297	0.6743	16.2729
	4.8925	5.0404	0.6953	16.3697
	7.0904	5.0636	82.7127	16.5736
	7.6592	5.0913	290.5997	16.7219
	7.6592	5.1121	810.9170	17.1179
	7.8173	5.1135	810.9179	17.1195
	7.8329	5.1272	810.9188	17.1670
	8.0296	5.1390	810.9190	17.3595
	8.1561	5.2663	810.9192	17.4819
	13.0978	5.3292	810.9194	17.8726
	13.4280	5.4068	810.9195	18.1687
	14.1469	5.7194	810.9201	18.3518
μ	4.1971	4.9218	228.8792	15.1777
σ	4.3204	0.3243	361.0539	2.0695

Table 4
t-tests (hypothesis testing for the difference in means of two samples) results for both kinds of robots between both methods (x = CMA-ES and y = NEAT).

	T-tests $\alpha = 0.05$	
	Planar robot	SCARA robot
t	−0.9162	3.2418
h	0 ($\mu_x < \mu_y$)	1 ($\mu_x > \mu_y$)
Significance	0.1835	0.0015

The results depicted in Table 4 show that the difference of the average MSE for both methods (CMA-ES and NEAT) for the planar robot is not statistically significant ($t = -0.9162, df = 29.3269, p < 0.05$) while for the SCARA robot the average MSE of the NEAT method is lower than that of the CMA-ES method ($t = 3.2418, df = 29.0019, p < 0.05$). However, this kind of comparison is not the most appropriate for comparing this kind of training methods since, in the general case, the system engineer is able to repeat the training process until good enough results are found. By this reason we have made another t-test including a selection of the 10 best runs of every method in every robot experiment. This comparison sounds more fair for judging the performance of a training method in this kind of problem.

In this way, the results in Table 5 (10 best runs) clearly show that the average MSE of the CMA-ES method is lower than that of the NEAT method for both kinds of robots: planar

Table 5
t-tests results for both kinds of robots between both methods (x = CMA-ES and y = NEAT) performed over the 10 best runs of each method.

	T-tests $\alpha = 0.05$ (best 10 runs)	
	Planar robot	SCARA robot
t	-44.5719	-20.6801
h	1 ($\mu_x < \mu_y$)	1 ($\mu_x < \mu_y$)
Significance	$9.7944e-20$	$3.3533e-009$

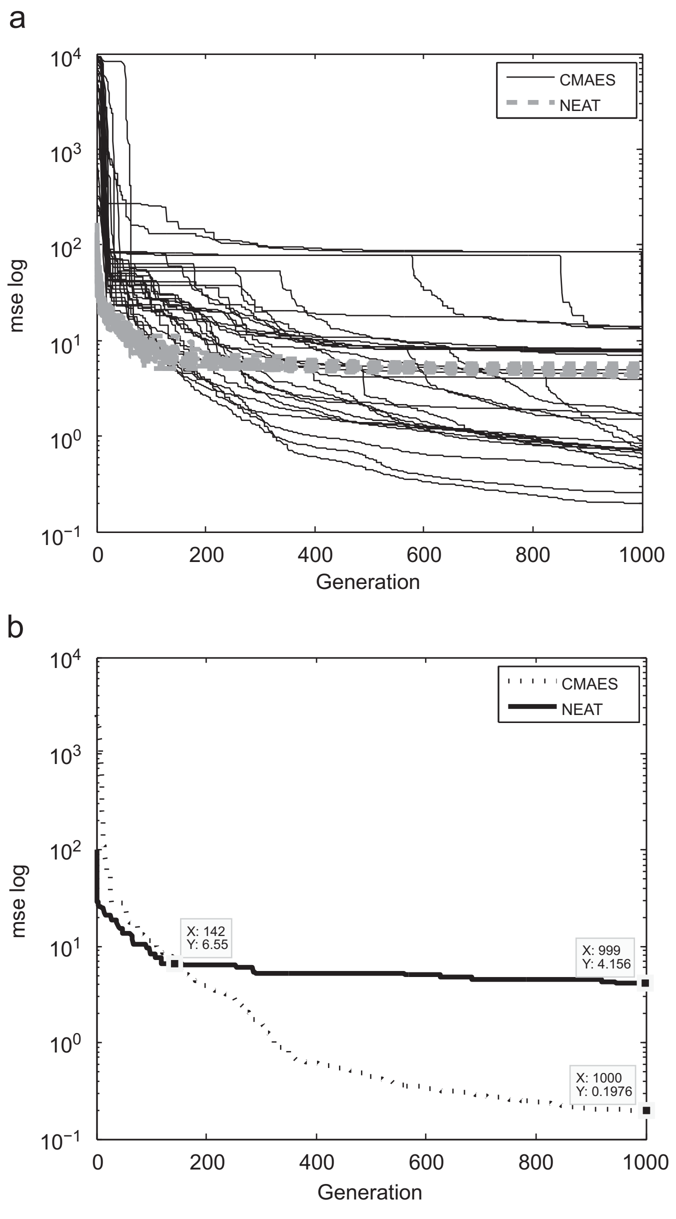


Fig. 5. Convergence of all experiments and best curves for the planar robot (logarithmic scale) over all the 30 runs.

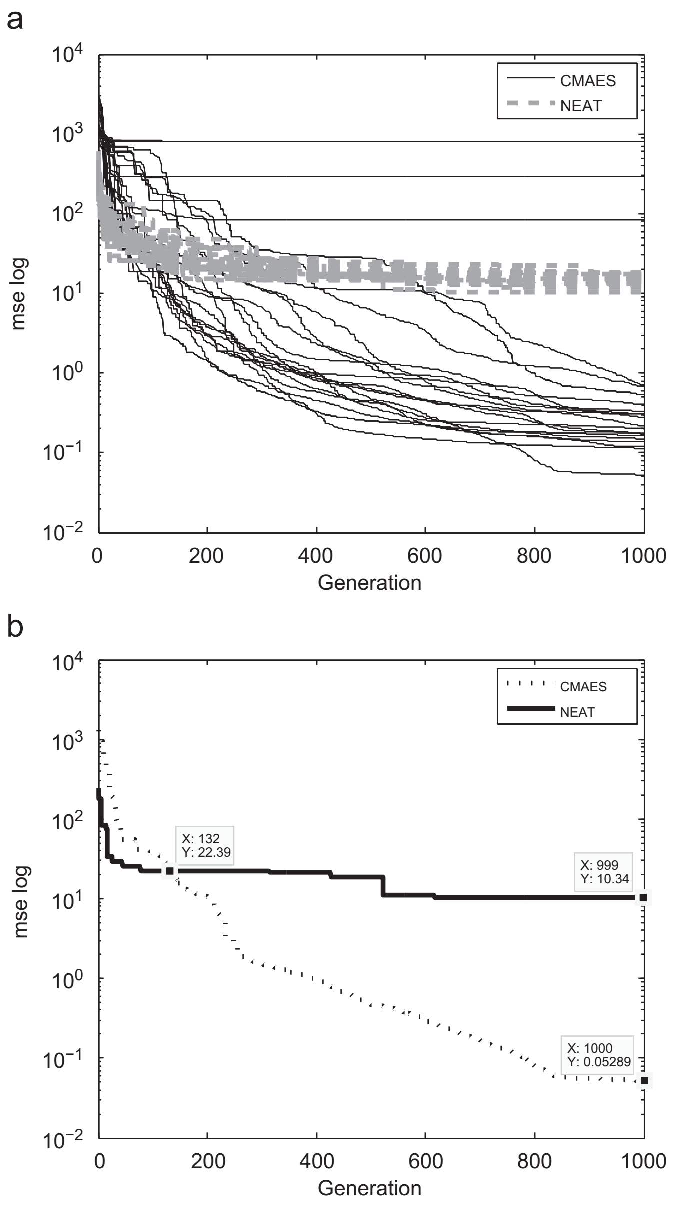


Fig. 6. Convergence of all experiments and best curves for the SCARA robot (logarithmic scale) over all the 30 runs.

($t = -44.5719, df = 17.4620, p < 0.05$) and SCARA ($t = -20.68018, df = 9.0052, p < 0.05$), respectively.

Figs. 5 and 6 show a graphical view of the results of the experimental work. The presented graphs show the behavior of the two methods (CMA-ES and NEAT) for each robot, so it can be easily seen the convergence of each method.

Generalization tests were carried out with new random target positions showing that the learned neuro-controllers generalize well over the whole space showing a deviation of 0.001% of the error goal during the learning process. This is consistent with our claim that it is a good strategy to train the neuro-controllers with a good representative set of fixed targets positions instead of variable target positions for the learning process that will introduce noise in the cost function and may result in poor convergence.

Fig. 5 shows the convergence curve of both methods for the problem of learning the inverse kinematics of the three-link-planar robot. As can be observed, CMA-ES with fixed topology surpasses the NEAT algorithm, producing, in the best cases, neuro-controllers with lower error than NEAT. One very relevant aspect is that NEAT was always able to find a good compromise solution in less time than CMA-ES. These results agree with the claims [15] that starting to learn with a reduced topology and adapt gradually the network topology to the task may increase the convergence speed.

Although NEAT has a good behavior during the first stages of the optimization problem, the CMA-ES algorithm is better developed for numerical optimization, which seems to be, mainly, the reason because of that after the generation where both strategies crosses as shown in Fig. 5 CMA-ES continues its adaptation and is able to find better solutions.

The general results for the planar robot show that the best error mark of the CMA-ES is $MSE = 0.1976$ while the best error mark of the NEAT algorithm was $MSE = 4.1564$, which is worse than the best CMA-ES error mark.

Fig. 6 shows the convergence curve of both methods for the problem of learning the inverse kinematics of the SCARA robot. As can be observed, the CMA-ES algorithm, with fixed topology, again surpasses the NEAT algorithm producing, in the best cases, neuro-controllers with lower error than NEAT. As we can see again, NEAT was able to find a good compromise solution in less time than CMA-ES, as in the previous experiment, but again when the lines of the two methods cross (for instance at generation 132 as in the bottom figure), CMA-ES starts to adapt better. The general results of the best case show that CMA-ES presents smaller quadratic error of $MSE = 0.0529$ than NEAT with a stationary error of $MSE = 10.3440$.

6. Conclusions and further work

A general method to learn the inverse kinematic of multi-link robots by means of neuro-controllers has been presented. The main advantage of the proposed method over traditional ones [1,3–5,9–13] is the simplicity of the training phase since it is not necessary to provide a set of valid input-target pairs in the training set but only a set of inputs without the need to know in advance a representative set of input-target pairs which in many problems could be unfeasible at all. Hence the main difference between input-target schemes and the method used here resides in a qualitative advantage in the complexity of the training requirements and not in the final results of the learned inverse kinematics.

Experimental results for two different robot models were also presented and evaluated. Such results indicate that the proposed method produces robust results in the problem addressed. Generalization tests were carried out with new random target positions showing that the learned neuro-controllers generalize well over the whole space showing a very little deviation of the error mark achieved during the learning process. This is consistent with our claim that it is a good strategy to train the neuro-controllers with a good representative set of fixed targets positions instead of variable target positions for the learning process that will introduce noise in the cost function and may result in poor convergence. Furthermore, the presented comparative analysis between structural and parametric evolutionary strategies suggests that the NEAT approach is a good alternative for complex problems where the network topology cannot be easily found by means of empirical tests. On the other hand, the CMA-ES method is clearly superior when an adequate topology is known in advance and when training with multiple startups or training attempts. One very relevant aspect is that NEAT was always able to find a good compromise solution in less time than CMA-ES and these results agree with the claims [15] that starting to learn with a reduced topology and adapt gradually the network topology to the task may increase the convergence speed. Although NEAT has a good behavior during the first stages of the optimization problem, the CMA-ES algorithm is better developed for numerical optimization, which seems to be, mainly, the reason because of that after the curves of both strategies crosses CMA-ES continues its adaptation and is able to find better solutions while NEAT enters in a stagnation state. The latter suggests that more theoretical and experimental work should be carried out in order to improve the parametric optimization scheme of the NEAT algorithm in order to improve its global performance for both parametric and structural optimization problems.

We observe that the general results of the work, although suggest a slightly better performance of one of the two algorithms under comparison, state that the proposed method is robust enough across different evolutionary algorithms.

Finally, regarding the main objectives of this research, and based on the exposed results, we have presented a method which is able to avoid any neural network learning algorithm which relies on the classical supervised input-target learning scheme since the applied method works only by specifying goal positions as the inputs without the requirement to provide a corresponding target.

To converge avoiding local optimal solutions. This point, although currently there is no theoretical way to prove such property, is well accepted in evolutionary computation since such methods are called global heuristic optimizers since they can explore the search space in a global way indeed being able to escape from local minima traps.

Finally, by means of neuro-evolutionary algorithms such as NEAT, the method was able to learn networks topologies while performing at the same time parametric learning finding good solutions.

Further lines of research include the development of a more sophisticated parametric learning mechanism for the NEAT algorithm in order to gain in convergence. Despite CMA-ES produces very good results, it is designed for being a general purpose optimization method and thus it lacks, which is normal, a way to perform topological learning in neural networks. It is in general not very difficult to find a good enough topology by trial and error and then apply a general optimization method such as CMA-ES. However, when performing online and in a truly unsupervised scenario, a complete neuro-evolutionary algorithm such as NEAT is desirable or indeed mandatory. Therefore, the future evaluation and comparison of new improved neuro-evolutionary algorithms with both parametric and topological learning capabilities against fixed topology learning algorithms as base lines should be a frequent matter of future research.

Acknowledgement

This work has been partially funded by the Spanish Ministry of Science and Technology, Project DPI2006-15346-C03-02.

References

[1] P.J. Alsina, N.S. Gehlot, Robot inverse kinematics: a modular neural network approach, in: Proceedings of 38th Midwest Symposium on Circuits and Systems, vol. 2, 1995, pp. 631–634.
[2] P.J. Angeline, G.M. Saunders, J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Transactions on Neural Networks 5 (1) (1994) 54–65.
[3] J. de Lope, T. Zarraonandia, R. González-Careaga, D. Maravall, Solving the inverse kinematics in humanoid robots: a neural approach, in: J. Mira, J.R. Álvarez (Eds.), Artificial Neural Nets Problem Solving Methods, vol. 2687, LNCS, Springer, Berlin, 2003, pp. 177–184.
[4] J. de Lope, D. Maravall, A biomimetic approach for the stability of biped robots, in: M.A. Armada, P. de González Santos (Eds.), Climbing and Walking Robots, Proceedings of the seventh International Conference CLAWAR 2004, Springer, Berlin, Heidelberg, 2005, pp. 593–600.
[5] J. de Lope Asiaín, R. González-Careaga, T. Zarraonandia, D. Maravall Gómez-Allende, Inverse kinematics for humanoid robots using artificial neural networks, in: R. Moreno-Díaz, F. Pichler (Eds.), Computer Aided Systems Theory—EUROCAST 2003, Revised Selected Papers, vol. 2809, LNCS, Springer, Berlin, 2003, pp. 448–459.
[6] J. Denavit, R. Hartenberg, A kinematic notation for lower-pair mechanisms based on matrices, ASME Journal of Applied Mechanics (1955), 215–221.

[7] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: International Conference on Evolutionary Computation, 1996, pp. 312–317.
[8] C. Igel, N. Hansen, S. Roth, Covariance matrix adaptation for multi-objective optimization, Evolutionary Computation 15 (1) (2007) 1–28.
[9] M.I. Jordan, D.E. Rumelhart, Forward models: supervised learning with a distal teacher, Cognitive Science 16 (1992), 307–354.
[10] Y. Kurematsu, T. Maeda, S. Kitamura, Autonomous trajectory generation of a biped locomotive robot using neuro oscillator, IEEE International Conference on Neural Networks II (1993) 1961–1966.
[11] T.M. Martinetz, H.J. Ritter, K.J. Schulten, Three-dimensional neural net for learning visuomotor coordination of a robot arm, IEEE Transactions on Neural Networks 1 (1) (1990) 131–136.
[12] A.S. Morris, A. Mansor, Finding the inverse kinematics of manipulator arm using artificial neural network with lookup table, Robotica 15 (1997), 617–625.
[13] J. Pereda, J. de Lope, D. Maravall, Comparative analysis of artificial neural network training methods for inverse kinematics learning, in: R. Marín, E. Onaindia, A. Bugarín, J. Santos (Eds.), Current Topics in Artificial Intelligence, vol. 4177, LNCS, Springer, Berlin, 2005, pp. 171–179.
[14] M.W. Spong, M. Vidyasagar, Robot Dynamics and Control, Wiley, New York, July 1989.
[15] K.O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evolutionary Computation 10 (2) (2002) 99–127.
[16] Y. Xia, J. Wang, A dual neural network for kinematic control of redundant robot manipulators, IEEE Transactions on Systems, Man, and Cybernetics, Part B 31 (1) (2001) 147–154.

José Antonio Martín H. received the B.S. and M.S. degrees in computer science from La Universidad del Zulia (LUZ) in 2002 and the Ph.D. degree in Computer Science from Universidad Politécnica de Madrid (UPM), in 2009, "Studies on Adaptive Systems with applications in Autonomous Robots and Intelligent Agents." In addition, he is on the Ph.D. program of "Fundamentals of Basic psychology" at the Universidad Nacional de Educación a Distancia (UNED), Madrid, where he has received the Advanced Studies Diploma on "A Computational Model of the Equivalence Class Formation Psychological Phenomenon". Since 2005 he has been with the Department of Informatic Systems and Computing of Universidad Complutense de Madrid (UCM). His main research areas are neuro dynamic programing, machine learning and cybemetics.

Javier de Lope (SM'94, M'98) received the M.Sc. in Computer Science from the Universidad Politécnica de Madrid in 1994 and the Ph.D. degree at the same university in 1998. Currently, he is an Associate Professor in the Department of Applied Intelligent Systems at the Universidad Politécnica de Madrid. His current research interest is centered on the study, design and construction of modular robots and multi-robot systems, and in the development of control systems based on soft-computing techniques. He is currently leading a three-year R&D project for developing industrial robotics mechanisms which follow the guidelines of multi-robot systems and reconfigurable robotics. In the past he also worked on projects related to the computer-aided automatic driving by means of external cameras and range sensors and the design and control of humanoid and flying robots.

Matilde Santos was born in Madrid, Spain. She received her B.Sc. and M.Sc. degrees in Physics Sciences (Computer Engineering) in 1984 and 1986, respectively, and her Ph.D. in Physics in 1994, from the Complutense University of Madrid (UCM). The Doctoral Dissertation was on "Commissioning Fuzzy Logic Controllers". Since 1986 she has been with the Department of Computer Architecture and Systems Engineering at the UCM, where she currently is a Senior Lecturer in System Engineering and Automatics. She is teaching several subjects in the Faculties of Physics and Computer Science at the UCM. She has directed and participated in several research projects and she has numerous scientific publications. Her major research interests are intelligent control (fuzzy, neuro-fuzzy, evolutive), process control, signal processing (machine learning, clustering, pattern recognition, SVM), modelling and simulation.