Engineering with Computers (2016) 32:553–565
DOI 10.1007/s00366-015-0432-z

ORIGINAL ARTICLE

A neuro-genetic-simulated annealing approach to the inverse kinematics solution of robots: a simulation based study

Raşit Köker[1] **· Tarık Çakar**[2]

Received: 16 April 2015 / Accepted: 23 December 2015 / Published online: 12 January 2016

Abstract In this study, a hybrid intelligent solution system including neural networks, genetic algorithms and simulated annealing has been proposed for the inverse kinematics solution of robotic manipulators. The main purpose of the proposed system is to decrease the end effector error of a neural network based inverse kinematics solution. In the designed hybrid intelligent system, simulated annealing algorithm has been used as a genetic operator to decrease the process time of the genetic algorithm to find the optimum solution. Obtained best solution from the neural network has been included in the initial solution of genetic algorithm with randomly produced solutions. The end effector error has been reduced micrometer levels after the implementation of the hybrid intelligent solution system.

Keywords Inverse kinematics solution · Genetic algorithms · Simulated annealing · Robotics · Hybrid intelligent system

1 Introduction

The inverse kinematics problem in robotics can be described as a mapping from the Cartesian space to the joint space. This problem is known as one of the most important issues in the field of robotics such as design, motion planning and control. Various approaches have been proposed by researchers for the solution of this problem, which, depends on the structural and workspace of the robot generally. The main task of a robot may be typically defined as tracking a desired trajectory in the Cartesian space with minimum error. In a real life application, robotic manipulators need joint variables for any given cartesian coordinate information to reach that point. The inverse kinematics solution is also used to obtain joint variables for any cartesian coordinate information [41]. Traditionally, there were three methods used to solve inverse kinematics problem of robotic manipulators, which are geometric [9, 26], algeabric [8, 10] and iterative methods [27]. Each method has some disadvantages for solving the inverse kinematics problem. For example, closed-form solutions are not guaranteed for the algebraic methods, and closed form solutions for the first three joints of the robot must exist geometrically when the geometric model is used. Similarly, the iterative inverse kinematics solution method converges to only one solution that depends on the starting point. These traditional solution methods may have a prohibitive computational cost due to the high complexity of the geometric structure of robots [22, 35]. That is the reason why researchers have focused on the solution of inverse kinematics problem by using intelligent techniques such as neural networks, genetic algorithms and etc.

Many researchers dealt with the neural-network-based inverse kinematics solution for robotic manipulators [1, 6, 11–13, 15, 17, 21, 28–30, 34, 40]. Tejomurtula and Kak [40] have suggested to use the structured neural networks, which can be trained quickly to overcome the disadvantages of the backpropagation algorithms, such as training time and accuracy for the inverse kinematics problem solution of a three-joint robotic manipulator. Karlik and Aydin

✉ Raşit Köker
rkoker@sakarya.edu.tr

Tarık Çakar
tcakar@sakarya.edu.tr

1 Electrical and Electronics Engineering Department, Faculty of Technology, Sakarya University, 54187 Sakarya, Turkey

2 Industrial Engineering Department, Engineering Faculty, Sakarya University, 54187 Sakarya, Turkey

[15] investigated the effect of different neural network configurations to identify the best configuration for a six-joint robot. They mentioned that six neural networks designed separately for each output with two hidden layers gives better solution comparing to a single neural network with six outputs. Oyama et al. [34] used modular neural network structures for learning in the inverse kinematics model. Their method is based on DeMers' method involving a number of experts, which are an expert selector, an expert generator, and a feedback controller that can accommodate the nonlinearities in the kinematic system. Their method contains certain limitations; for instance, the inverse kinematics computation procedure is highly complex, and the learning speed is low. However, root mean square (RMS) hand position errors of less than 10 mm were observed. Mayorga and Sanongboon [30] proposed a novel neural-network approach for fast inverse kinematics computations and efficient singularity avoidance or prevention for redundant robots based on a set of bounded geometrical concepts used to decide the characteristic matrices. Their algorithm activates the realization of fast and robust real-time algorithms for safe robot path planning. Bingul and Ertunc [1] presented a study based on a neural network approach based on using backpropagation algorithm for the inverse kinematics solution of an industrial robotic manipulator without an analytical inverse kinematics solution. The large errors in the joint angles and the inability of this approach to provide multiple solutions to the inverse kinematics problem have been reported as the disadvantages of their approach. Köker [17] presented a study based on using parallel neural networks for the inverse kinematics solution of a six-joint robot model. The best neural network result has been selected among the obtained solutions. Two studies have been presented by Hasan et al. [12, 13] regarding inverse kinematics solution for a 6-DOF robotic manipulator. In the first paper [12], to control the motion of a 6-DOF robotic manipulator and to overcome difficulties in solving the inverse kinematics problem such as uncertainties and singularities an adaptive learning strategy using an artificial neural network was proposed. In their approach, a neural network was trained to teach a desired set of joint angle positions from a set of given end effector positions, and the experimental results denoted good mapping over the working area of the robot. The researchers also provided a graphical presentation of these errors by iteration number. Their a second paper [13] was based on using artificial neural networks to learn robot system characteristics rather than specifying an explicit robot system model to overcome singularities and uncertainties; this method was implemented for another type of 6-DOF robotic manipulator model. Additionally, some researchers have focused on using genetic algorithms [7, 14, 32, 33, 39] and artificial bees algorithm [36]. These algorithms are known as optimization algorithms and take more process because of search based working.

This paper has focused on the improvement of the precision of the inverse kinematics solution obtained from a neural network using the improved hybrid intelligent algorithm that includes neural network, genetic algorithm and simulated annealing algorithm. A neural network was trained to obtain inverse kinematics solutions for a 4-DOF robotic manipulator. The aim of first training a neural network was to obtain a solution to put into the initial population of the genetic algorithm. In the genetic algorithm, the floating-point portion of the solution was improved up to ten significant digits. The neural network prediction results and the randomly generated solutions were placed in the initial population of the genetic algorithm. In the proposed solution system, SA is used as a genetic operator. The best solution of the each population is given to SA as an initial solution. When the SA found the first better solution than the initial, obtained solution is given to GA to generate the new population. Solution method is given below in Fig. 2. Thus, it is provided to decrease as much as possible the number of repetition of the best solution alternative obtained from the genetic algorithm in each stage. Since SA is trying to find a better solution than GA's, the performance of the genetic algorithm will be more increased. In the graphic, solution values obtained using both only GA and GA+SA hybrid solution system have been demonstrated. During the implementation of the genetic algorithm, the direct kinematics equations of the robotic manipulator were used to compute the end effector position error. On the other hand, SA was used as a genetic operator in the proposed solution scheme to reduce the number of generation to reach the optimum solution. This error in the end effector of the robotic manipulator was used as a fitness function in the genetic algorithm. The main contribution of this study is improvement of the precision of the solution obtained from the neural network. The results presented show that the precision of the inverse kinematics solution was highly and significantly improved to micrometer levels. This solution approach can be used in various critical robotic applications such as micro-assembly, cutting, drilling, and medical operations.

2 Kinematics analysis of robotic manipulators

Robotic manipulators and kinematic mechanisms are typically constructed by connecting different joints together using rigid links. A robot can be modeled as an open-loop

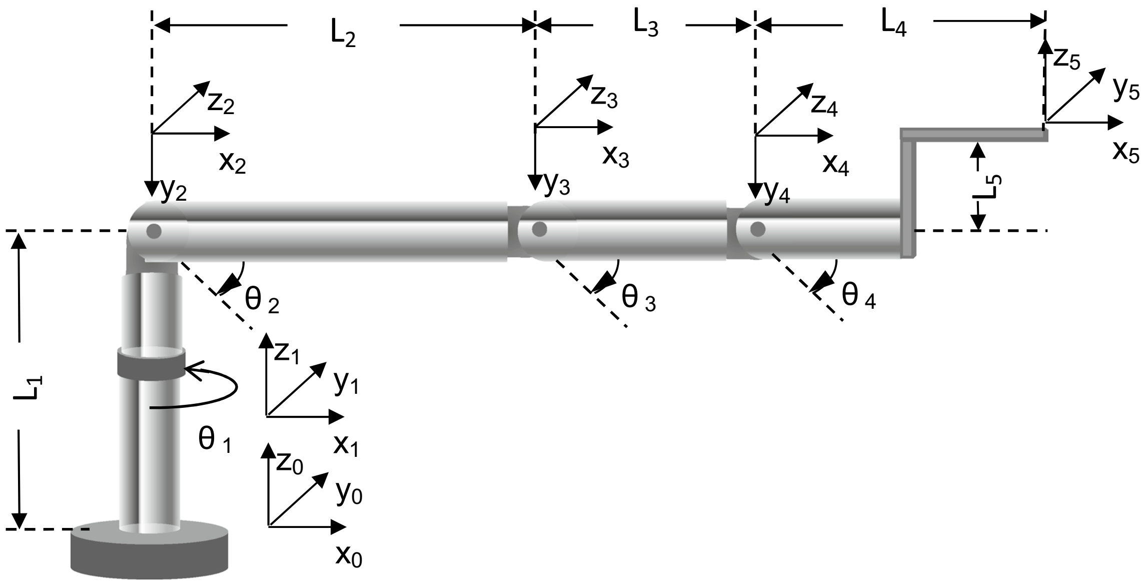


Fig. 1 The structure of robot model

Table 1 *D–H* parameters of the robot

i	a_{i-1} (mm)	α_{i-1} (°)	d_i (mm)	θ_i (°)
1	0	0	$L_1 = 80$	$-90 < \theta_1 < 90$
2	0	−90	0	$-180 < \theta_2 < 0$
3	$L_2 = 120$	0	0	$0 < \theta_3 < 145$
4	$L_3 = 65$	0	0	$-90 < \theta_4 < 90$
5	$L_4 = 85$	90	$L_5 = 25$	$\theta_5 = 0$

articulated chain with these several rigid links connected in series by either revolving or prismatic joints driven by actuators. Robot kinematics deals with the analytical study of the geometry of motion of a robot with respect to a fixed reference coordinate system as a function of time, without regard to the forces or moments that cause the motion [24, 25]. For this reason, it deals with the analytical description of the robot as a function of time, particularly the relations between the joint-variable space and the position and orientation of the end effector of a robotic manipulator [10, 40]. Figure 1 shows the structure of the robotic manipulator used in this study. In addition, the Denavit–Hartenberg parameters of used four-DOF robot are given in Table 1 [42].

A four-DOF robot model has been used in this study, as shown in Fig. 1. A 3-D view of the robot model is given in Fig. 2. By using the above notations and Fig. 1, the *D–H* parameters are obtained for this robot model as given in Table 1 regarding four parameters, which can be defined as a_{i-1}, α_{i-1}, d_i, and θ_i, that are the link length, twist, offset and angle of joint, respectively.

$$ {}_{i}^{i-1}T = (\propto_{i-1}) D_x(a_{i-1}) R_z(\theta_i) Q_i(d_i) \tag{1}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x \begin{bmatrix} c_i & -s_i & 0 & 0 \\ s_i & c_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$= \begin{bmatrix} c_i & -s_i & 0 & a_{i-1} \\ s_i c\alpha_{i-1} & c_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s_i s\alpha_{i-1} & c_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

In (1), (2) and (3), R_x and R_z refer to rotation, D_x and Q_i refer to translation, and c_i, s_i, $c\alpha_{i-1}$ and $s\alpha_{i-1}$ can be explained as the short-hands for $\cos\theta_i$, $\sin\theta_i$, $\cos\alpha_{i-1}$ and $\sin\alpha_{i-1}$, respectively.

For the four-DOF robot, each of the link transformation matrices using (1) can be calculated as given in the following form;

$$ {}_{1}^{0}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

$$ {}_{2}^{1}T = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5}$$

$$ {}^{2}_{3}T = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (6) $$

$$ {}^{3}_{4}T = \begin{bmatrix} c_4 & -s_4 & 0 & L_3 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (7) $$

$$ {}^{4}_{5}T = \begin{bmatrix} 1 & 0 & 0 & L_4 \\ 0 & 0 & -1 & -L_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (8) $$

The forward kinematics of the end effector based on the base frame can be determined by the multiplication of all of the ${}^{i-1}_{i}T$ matrices as given below;

$$ {}^{b}_{e}T = {}^{0}_{1}T {}^{1}_{2}T {}^{2}_{3}T \ldots {}^{i-2}_{i-1}T {}^{i-1}_{i}T. \quad (9) $$

Another expression of ${}^{b}_{e}T$ can be expressed as;

$$ {}^{b}_{e}T = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (10) $$

In Eq. (10), p_x, p_y, and p_z denotes the elements of the position vector whereas n_x, n_y, n_z, s_x, s_y, s_z, a_x, a_y, a_z denote the rotational elements of the transformation matrix.

The position and orientation of the end effector based on the base is given in (11) for a four-jointed manipulator [23, 37].

$$ {}^{0}_{5}T = {}^{0}_{1}T {}^{1}_{2}T {}^{2}_{3}T {}^{3}_{4}T {}^{4}_{5}T. \quad (11) $$

where,

$$ n_x = c_1c_2c_3c_4 - c_1s_2s_3c_4 - c_1c_2s_3s_4 - c_1s_2c_3s_4 \quad (12) $$

$$ n_y = s_1c_2c_3c_4 - s_1s_2s_3c_4 - s_1c_2s_3s_4 - s_1s_2c_3s_4 \quad (13) $$

$$ n_z = -s_2c_3c_4 - c_2s_3c_4 + s_2s_3s_4 - c_2c_3s_4 \quad (14) $$

$$ s_x = -s_1 \quad (15) $$

$$ s_y = c_1 \quad (16) $$

$$ s_z = 0 \quad (17) $$

$$ a_x = c_1c_2c_3s_4 - c_1s_2s_3s_4 + c_1c_2s_3c_4 + c_1s_2c_3c_4 \quad (18) $$

$$ a_y = s_1c_2c_3s_4 - s_1s_2s_3s_4 + s_1c_2s_3c_4 + s_1s_2c_3c_4 \quad (19) $$

$$ a_z = -s_2c_3s_4 - c_2s_3s_4 - s_2s_3c_4 + c_2c_3c_4 \quad (20) $$

$$ \begin{aligned} p_x = {} & L_4c_1c_2c_3c_4 - L_4c_1s_2s_3c_4 - L_4c_1s_3s_4 - L_4c_1s_2c_3s_4 \\ & + L_5c_1c_2c_3s_4 - L_5c_1s_2s_3s_4 + L_5c_1c_2s_3c_4 \\ & + L_5c_1s_2c_3c_4 + L_3c_1c_2c_3 - L_3c_1s_2s_3 + L_2c_1c_2 \end{aligned} \quad (21) $$

$$ \begin{aligned} p_y = {} & L_4s_1c_2c_3c_4 - L_4s_1s_2s_3c_4 - L_4s_1c_2s_3s_4 - L_4s_1s_2c_3s_4 \\ & + L_5s_1c_2c_3s_4 - L_5s_1s_2s_3s_4 + L_5s_1c_2s_3c_4 \\ & + L_5s_1s_2c_3c_4 + L_3s_1c_2c_3 - L_3s_1s_2s_3 + L_2s_1c_2 \end{aligned} \quad (22) $$

$$ \begin{aligned} p_z = {} & -L_4s_2c_3c_4 - L_4c_2s_3c_4 + L_4s_2s_3s_4 - L_4c_2c_3s_4 \\ & - L_5s_2c_3s_4 - L_5c_2s_3s_4 - L_5s_2s_3c_4 + L_5c_2c_4 \\ & - L_3s_2c_3 - L_3c_2s_3 - L_2s_2 + L_1 \end{aligned} \quad (23) $$

After the computations all obtained notations based on Eq. (11), have been given above.

To train the neural network, the training and test sets have been prepared by using Eqs. (12)–(23) for the inverse kinematics model learning.

3 Proposed hybrid intelligent solution system

In the proposed solution system, SA is used as a genetic operator. The best solution of the each population is given to SA as an initial solution. When the SA found the first better solution than the initial, obtained solution is given to GA to generate the new population. Solution method is given below in Fig. 2. Thus, it is provided to decrease as much as possible the number of repetition of the best solution alternative obtained from the genetic algorithm in each stage. Since SA is trying to find a better solution than GA's, the performance of the genetic algorithm will be more increased. In the graphic, solution values obtained using both only GA and GA + SA hybrid solution system have been demonstrated.

3.1 Neural network implementation

In the neural network based inverse kinematics solution part, a backpropagation neural network with sigmoidal activation function has been designed. The Eqs. 1–11 have been used for the preparation of the training and test sets, which have been presented in the kinematic analysis section 8000 data has been prepared and 6000 of them has been used to form training set and the remaining part has been used to form test set. It has been completed approximately in 476 epochs. The neural network is designed including 46 perceptrons in hidden layer, and 12 perceptrons in input and 4 perceptrons in output layer. The momentum rate (α) and learning rate (η) have been determined experimentally as 0.85 and 0.35, respectively. The number of perceptrons in hidden layer has also been determined experimentally. Observed error at the end of the learning is 0.00098 for training set.

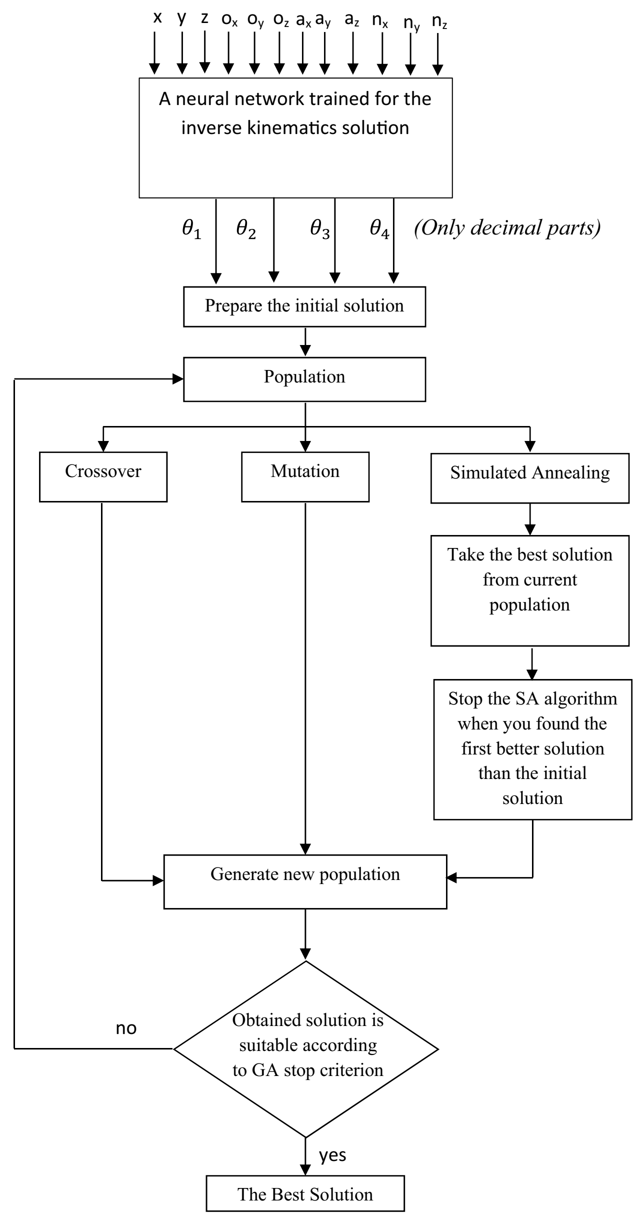

x y z o$_x$ o$_y$ o$_z$ a$_x$ a$_y$ a$_z$ n$_x$ n$_y$ n$_z$
A neural network trained for the inverse kinematics solution
θ_1 θ_2 θ_3 θ_4 (Only decimal parts)
Prepare the initial solution
Population
Crossover
Mutation
Simulated Annealing
Take the best solution from current population
Stop the SA algorithm when you found the first better solution than the initial solution
Generate new population
Obtained solution is suitable according to GA stop criterion
no
yes
The Best Solution

Fig. 2 The proposed hybrid intelligent solution system

3.2 Genetic algorithms

In this part, the genetic algorithms, which are an adaptive system tool and derived from evolution theory, have been explained. Basically, to comprise the population which better solutions are evolved over a series of generations a number of solutions are generated. First, the coding of the problem is needed with the condition that it should be fitting with the GA. GA operators are applied on chromosomes after the implementation of coding. The obtained new offsprings are not guaranteed to be good solutions by the working of crossover and mutation operators until an optimal solution is obtained, crossover, mutation, and reproduction processes will continue. In this paper, the case solutions shows the solutions that give less end effector position error, which has been defined as a fitness function. [18, 19, 31]. The implementation of genetic algorithms have been given below with numerical examples.

3.2.1 Coding

First the best solution has been obtained by the designed neural network and then the floating point part of this solution is converted to binary. To avoid from unnecessary process time, only ten digits of the floating point part of the solution is used in the implementation of GA. Additionally, during the coding process the maximum possible value regarding the problem space is used in the calculations to get information about the maximum possible number of digit in the solutions.

The obtained solutions from the neural network as angular position information are used by the robot to reach the target point, which is known as cartesian position information. However, the solutions obtained by the neural network may have some error due to the characteristic of neural network. A neural network can produce a solution with acceptable error. It can be trained until the error is found acceptable. Especially, floating point parts are significant in the training of a neural network to get the best result.

For instance, the obtained angle values from ANN are as presented below;

Obtained result from the ANN block	Floating number part	Binary representation of the floating part
1. Angle: 22.5180007805	5180007805	010011010011000000 1010010101111101
2. Angle: −98.3451961106	3451961106	001100110111000000 1011111100010010
3. Angle: 83.8671597631	8671597631	100000010011011110 0001010000111111
4. Angle: 16.9206805780	9206805780	100010010011000100 1011010100010100

3.2.2 Initial population

The full of the initial population is not produced randomly. Obtained best solution from the neural network part is also included in the population. Other chromosomes are generated randomly. The chromosomes are generated randomly based on taking care of maximum number of digit in the solution space. In the generation of the initial population 34 digits are used.

For instance, a sample randomly generated values for each joint in the initial population is supposed as presented below:

	Randomly generated binary number	The decimal representation of the generated binary number
Chromosome #1	1000001110011110001010110111010101	8832724437
	1001000010010010100010010011110101	9702089973
	0001001010000010111010110011110000	1242279152
	0000001011110011101011101001100100	198097508
Chromosome #2	0101011010110110100110111110111011	5819232187
	0111010001111110100110001010100011	7817814691
	1111000011001010110111000110100111	16159306151
	0101001111011010100011101111110111	5627329527

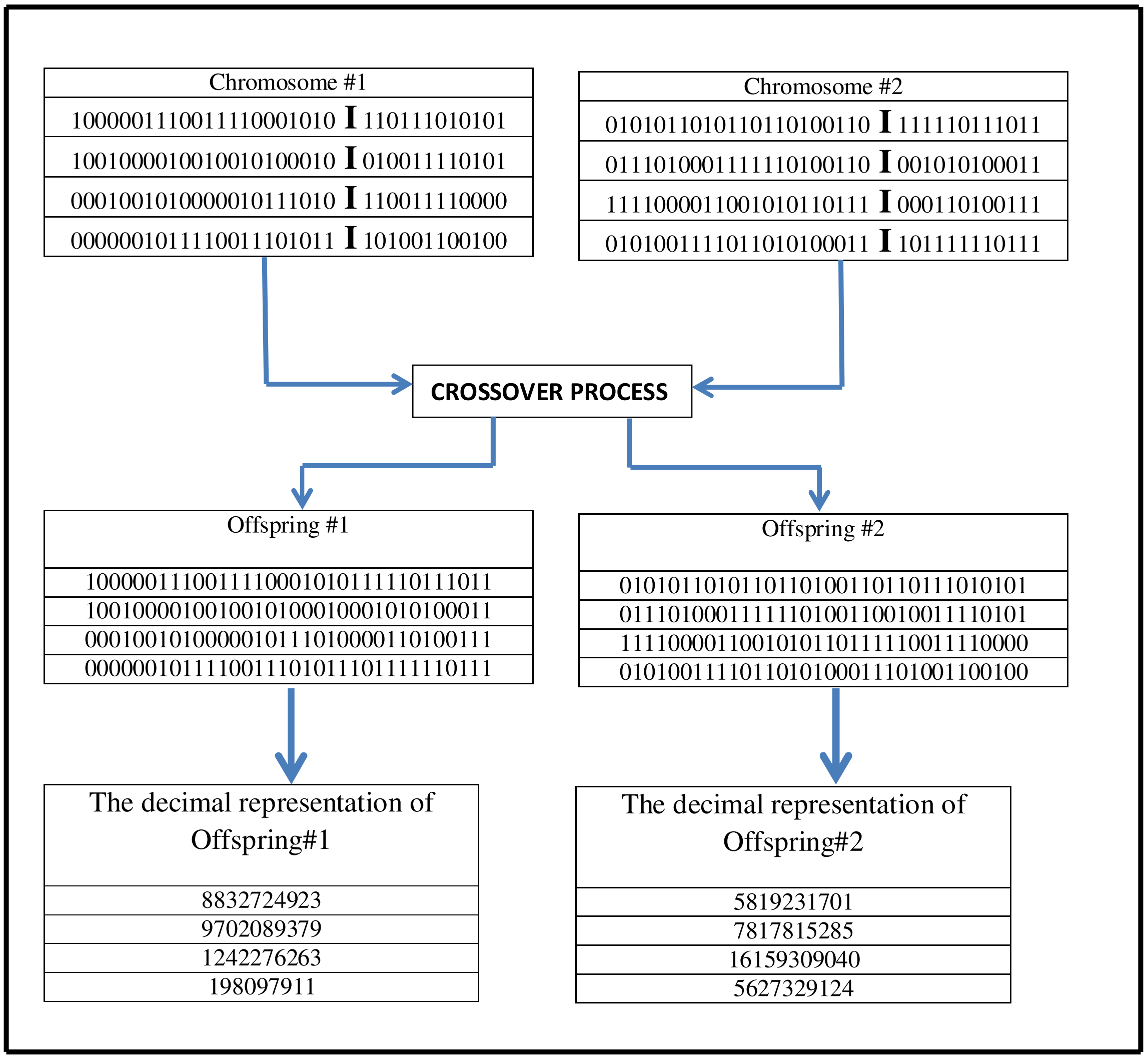
3.2.3 Crossover

The crossover process is done based on crossing obliquely from a cut point that is randomly determined two chromosomes. Two new chromosomes will be obtained after the implementation of this process. By the way of crossing these two chromosomes to each other, the new binary numbers referring to the floating part of the robot joint angle will be gotten.

Below a sample implementation of the crossover operation has been presented. Each related part of the chromosomes will be crossed with the related part of the other chromosome. For instance, the first part of the first chromosome will be crossed with the first part of the second chromosome. The same operation will be done for all chromosomes. These processes are given below showing chromosomes and crossover points.

	Define randomly the crossover point and crossover process	The decimal representation of the generated binary number
Chromosome #1	1000001110011110001010 **I** 110111010101	8832724437
	1001000010010010100010 **I** 010011110101	9702089973
	0001001010000010111010 **I** 110011110000	1242279152
	0000001011110011101011 **I** 101001100100	198097508
Chromosome #2	0101011010110110100110 **I** 111110111011	5819232187
	0111010001111110100110 **I** 001010100011	7817814691
	1111000011001010110111 **I** 000110100111	16159306151
	0101001111011010100011 **I** 101111110111	5627329527

The Crossover process has been shown below;



Obtained new offsprings in the binary form are converted to the decimal form to become new floating number of the neural network result. The new values after crossover process have been shown below.

New result—1	New result—2
22.8832724923	22.5819231701
−98.9702089379	−98.7817815285
83.1242276263	83.16159309040
16.198097911	16.5627329124

3.2.4 Mutation

In the mutation process, a gene is randomly selected from the chromosomes in the population based on the defined mutation rate. In this study, this gene will be a bit inside the binary numbers in the chromosome. The randomly selected genes from the population are mutated, it means if the gene is 0, it is made 1, if the gene is 0 it is made 1. By this operation the new offsprings are obtained.

There are 272 genes in the population, the genes are numbered between 1 and 272 and randomly, the randomly selected genes are mutated. Since mutation rate has been defined as 1 %, 272 × 0.01 = 2.72 ~ 3 genes have been randomly selected and mutated.

An example of mutation operation is presented below:

	Randomly generated binary number	The decimal representation of the generated binary number
Chromosome#1	10000011100111100 01010110111010101	8832724437
	10010000100100101 00010010011110101	9702089973
	00010**0**101000001011 1010110011110000	1242279152
	000000101111001110 1011101001100100	198097508

	Randomly generated binary number	The decimal representation of the generated binary number
Chromosome#2	010101101011011010 0110111**1**0111011	5819232187
	011101000111111010 0110001010100011	7817814691
	111100001100101011 0111000110100111	16159306151
	01010011110**1**101010 0011101111110111	5627329527

The obtained new offsprings after mutation process have been shown below.

	New offsprings	The decimal representation of the generated binary number
Offspring #1	100000111001111000 1010110111010101	8832724437
	100100001001001010 0010010011110101	9702089973
	00010**1**101000001011 1010110011110000	**1510714608**
	000000101111001110 1011101001100100	198097508
Offspring #2	010101101011011010 0110111**0**0111011	**5819232059**
	011101000111111010 0110001010100011	7817814691
	111100001100101011 0111000110100111	16159306151
	01010011110**0**101010 0011101111110111	**5623135223**

As mentioned above the error between the target point and end effector of the robot is defined as the fitness function in this study since the main aim of this paper is to minimize the error. The other parameters used during the implementation of genetic algorithms are given below:

$$\begin{aligned}&\text{Population size}: \ 100, \ \text{Crossover rate}: \ \%100\\&\text{Mutation rate}: \ \%1, \quad \text{Max generation}: \ 150\end{aligned}$$

Parameters are defined using trial processing. For example to define population size first used 50 population, after 75 and 100 population. To obtain better solutions and faster processing time 100 population more suitable. Also different percentage of crossover and mutation rate has been tried after most efficient rates have been obtained.

3.2.5 Reproduction

By reproduction operator, a copy of each gene is gotten and added to the list of candidate genes during the population. Fundamentally, this will guarantee that each chromosome in the current population remains a candidate to be selected for the next population. It is aimed to find solution minimizing the given fitness function in this paper. Here, the fitness function is the position error as a distance between the robot end effector and the target as it is stated above. The genetic algorithm may get better chances to survive chromosomes with quite higher fitness. The good chromosomes will stay in the population as living ones. This process will keep going until an optimal solution is taken.

4 Simulated annealing algorithm

The SA algorithm is a metaheuristic technique for the solution of the combinatorial optimization problems based on the physical phenomenon of annealing. In the literature, it was first suggested by Kirkpatrick et al. [16]. The solution of a combinatorial optimization problem can be done by using SA in a manner that is analogous to the process of annealing. The algorithm is based on two results from statistical physics, namely, the probability of a system getting a given energy E at thermodynamic balance, and the so-called Metropolis algorithm that may be used in the simulation of the evolution of a system towards thermodynamic balance at a given temperature. A control parameter is introduced to simulate the temperature of the system. The number of accessible energy states are controlled by the temperature. And the temperature is leading to a locally optimal state in case of lowered gradually. In the system the energy resembles the objective function value in a minimization problem, while a feasible solution resembles a certain state of the system. The final solution resembles to the system getting frozen in its ground state [38] (Fig. 3).

The flowchart of the SA algorithm has been given in Fig. 1 [20]. The SA starts with an initial solution (A), initial temperature (T) and iteration number (C) as denoted in the figure. The possibility of the acceptance of the disturbing solution is controlled by the temperature as mentioned above. But, the reason of using the iteration number is to determine the number of repetitions until that a solution is found on a stable state under the temperature [2, 3]. The temperature gets the following implicit flexibility index meaning. At the beginning of the searches, it means at high temperature situation, some flexibility may be moving to a worse solution cases, on the other hand, less of this flexibility exists in the searches done later that means at lower temperature. A new neighborhood solution (N) will be generated based on these T, C through a heuristic perturbation on the existing solutions. When the improvement on the change of an objective function has been observed, the neighborhood solution (N) will be seen as a good solution. Even if the change of an objective

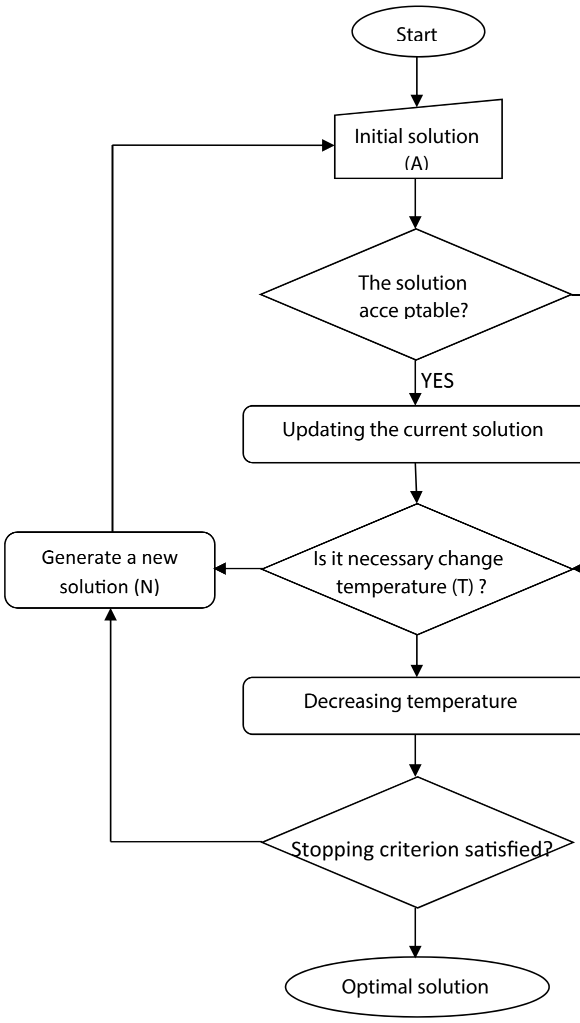

Start
Initial solution (A)
The solution acce ptable?
YES
Updating the current solution
Is it necessary change temperature (T) ?
Generate a new solution (N)
Decreasing temperature
Stopping criterion satisfied?
Optimal solution

Fig. 3 Flowchart of the simulating algorithm

function is not improved, the neighborhood solution will be a new solution with a convenient probability, which is based on $e^{-D/T}$. This situation removes the possibility to find a global optimum solution out of a local optimum. If there is not any change after certain iterations, the algorithm will be stopped. The algorithm goes on with a new temperature value, if there is still improvement on the new solution.

To apply SA to practical problems, there are several factors to be determined at the beginning. The definition of a procedure to generate neighborhood solutions from a current solution is needed, first. Some parameters should be appropriately determined for the generation of these solutions efficiently. An initial temperature, the number of repetitions, conditions for completion and the ratio of temperature change can be given as examples for these parameters. To get a good solution the combination of these parameters should be arranged according to the problem [4, 5].

4.1 Coding

The floating points, part of the solution obtained from the neural network have been converted to binary form as 10 digits. To get robot to go to target point the angular position information obtained from the neural network is used. These angular position information are obtained for any given cartesian position. But, these obtained angular values may still have some errors since a neural network can learn with an acceptable error. In other words, there will be a position error at the end effector by using obtained angular position information from the neural network.

For instance, the neural network results obtained for each joint have been presented below;

Obtained result from the ANN block	Floating number part	Binary representation of the floating part
1. Angle: 39.**5594683728**	5594683728	0101001101011110000001100101010000
2. Angle: −72.**12393023574**	12393023574	1100011000010100110111000101010110
3. Angle: 64.**6219724124**	6219724124	0101110010101110010111010101011100
4. Angle: −21.**9083836235**	9083836235	1000011101011100000101011101001011

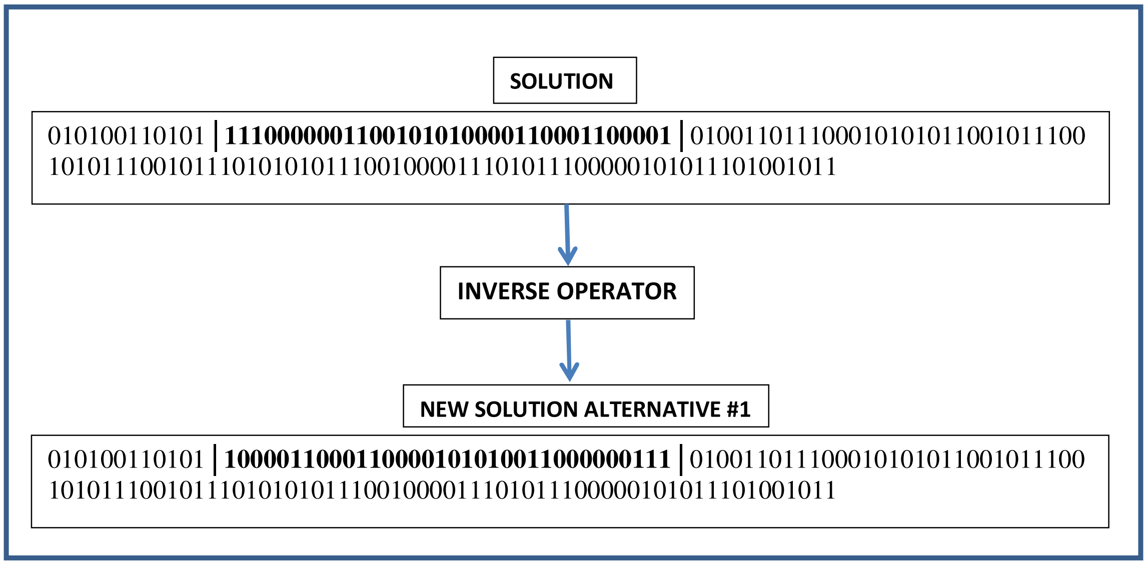
The initial solution of SA has been constituted by using the obtained solution from the neural network. Floating point part of the solution for each joint has been coded in the binary form. SA has been applied to floating points part of the solutions for each joint. Binary numbers have been combined with each other according to the angle sequence. Therefore, the first solution has been coded to be used in SA as an initial solution in the following form;

Binary representation of the floating part	Combined binary floating part for initial solution of simulated annealing
0101001101011110000001100101010000	0101001101011110000001100101010000110001100001010011011100010101011001011100101011100101110101010111001000011101011100000101011101001011
1100011000010100110111000101010110	
0101110010101110010111010101011100	
1000011101011100000101011101001011	

It can be applied two kind of operator to generate new solution in SA.

1. Operator: Inverse operator

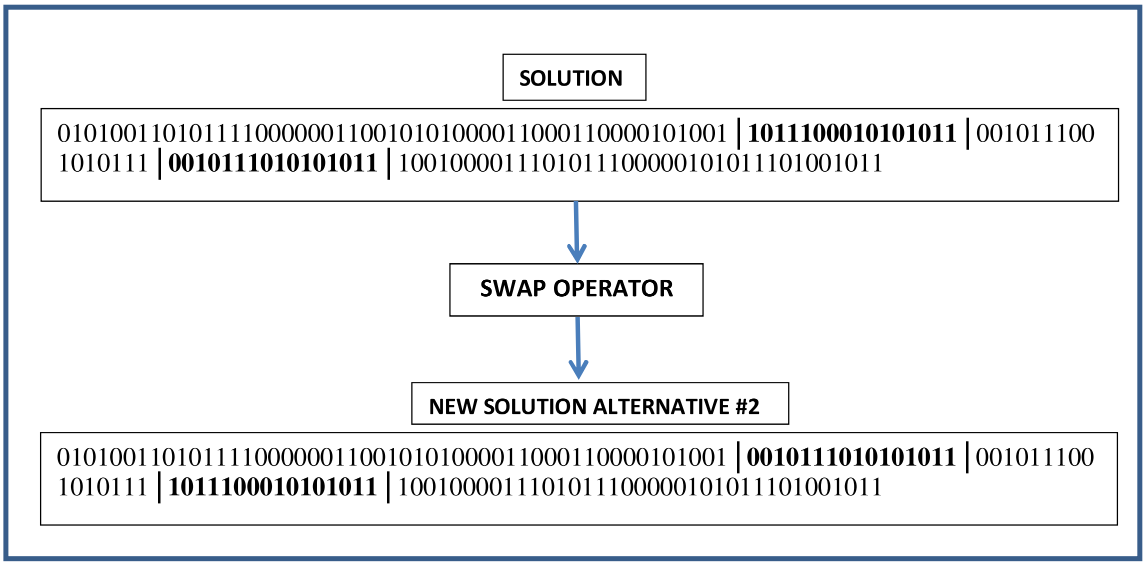
Two points are selected in the current solution and applied the inverse operator to obtain new solution alternative using inverse operator as given in the following;



New solution alternative #1 for floating part	New solution alternative
010100110101100001100011000 0101010	1. Angle : 39.**5593205820**
011000000111010011011100010 1010110	2. Angle : -72.**6473085270**
010111001010111001011101010 1011100	3. Angle: 64.**6219724124**
100001110101110000010101110 1001011	4. Angle : -21.**9083836235**

2. Operator: Swap operator

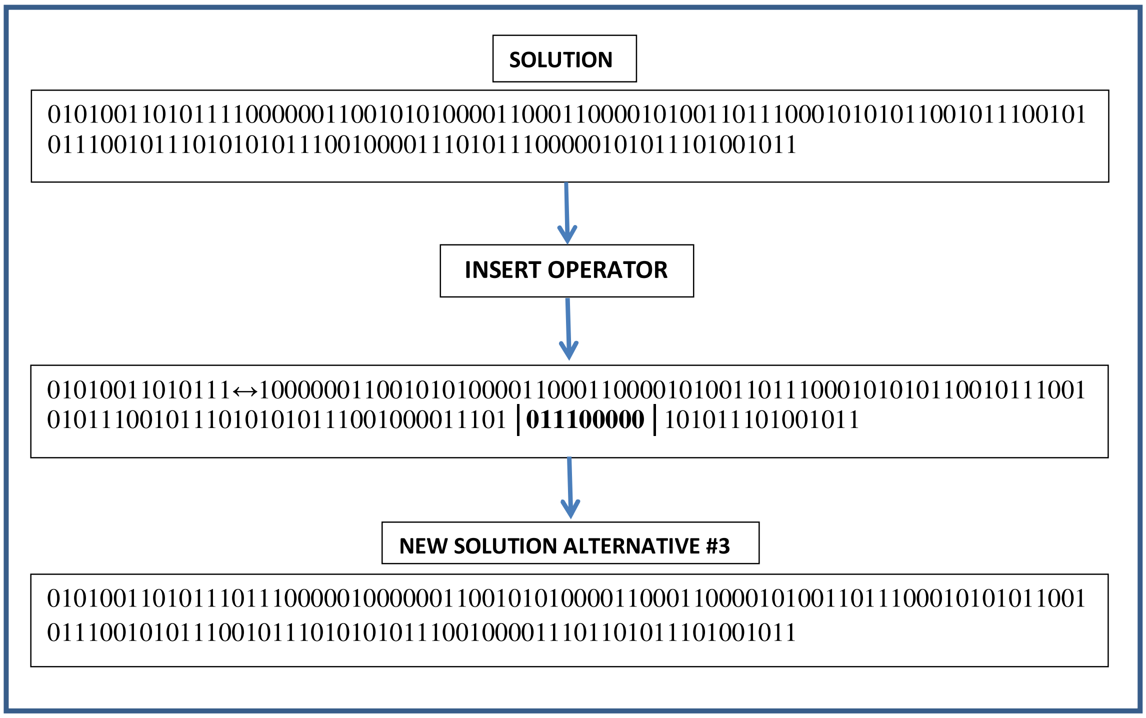
Two pieces of current solution are selected randomly, and swapped as follow:



New solution alternative #1 for floating part	New solution alternative
0101001101011110000001100101010000	1. Angle: 39.**5594683728**
1100011000010100100101110101010110	2. Angle: −72.**13292952918**
0101110010101111011100010101011100	3. Angle: 64.**6220006748**
1000011101011100000101011101001011	4. Angle: −21.**9083836235**

3. Operator: Insert operator

First, a piece of numbers is selected randomly, which insert to the randomly selected point;



01010011010111110000001100101010000110001100001010011011100010101011001011100101
01110010111010101011100100001110101110000010101110100101 1

0101001101011 1↔100000011001010100001100011000010100110111000101010110010111001
01011100101110101010111001000011101 | **011100000** | 101011101001011

01010011010111011100000100000011001010100001100011000010100110111000101010 11001
0111001010111001011101010101110010000111011010111010 01011

New solution alternative #3;

New solution alternative #1 for floating part	New solution alternative
0101001101011101110000010000001100	1. Angle: 39.**5594612748**
1010100001100011000010100110111000	2. Angle: −72.**1130025208**
1010101100101110010101110010111010	3. Angle: 64.**11487763642**
1010111001000011101101011101001011	4. Angle: −21.**11694692171**

As it has been explained above, the fitness function is defined as the error between the target point and end effector of the robot. The main purpose of the paper is to minimize this error. Long running time and difficulty in selecting cooling parameter when the problem size becomes larger are the weak points of SA. A geometric ratio has been used in SA as $T_{k+1} = \alpha T_k$, where T_k and T_{k+1} are the temperature values for k and $k + 1$ steps, respectively. Geometric ratio is used more commonly in practice. In this study, the initial temperature was taken 10,000 and 0.95 was used for cooling ratio (α).

5 Results and discussions

In this study, a hybrid intelligent solution approach including neural networks, genetic algorithm and simulated annealing has been applied to solve the inverse kinematics problem of a four-joint robotic manipulator. The proposed approach is based on training of the neural networks towards the inverse kinematics solution and improving the neural network results by using genetic algorithm and simulated annealing module. By this way, the error at the end effector at the end of training has been minimized for some

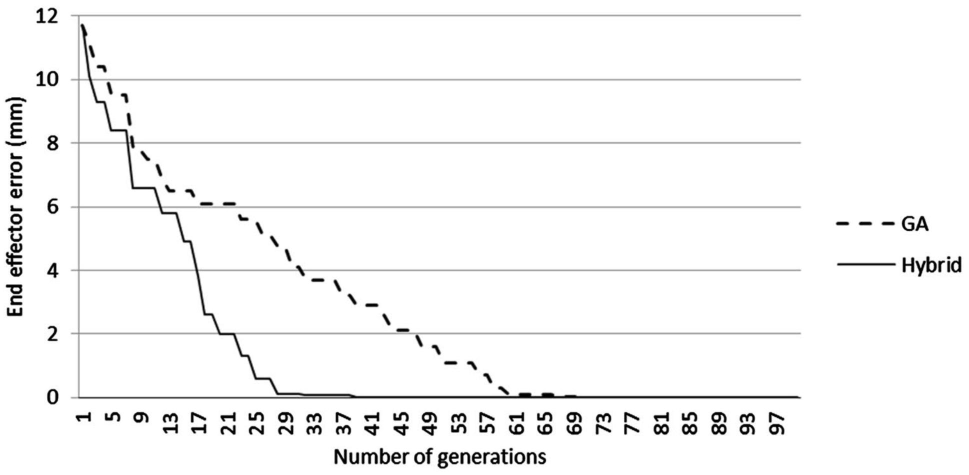


Fig. 4 The graphic of error change versus number of generations for GA and hybrid model

critical applications. A neural network has been first trained to obtain a solution to be included in the initial solution pool of genetic algorithm. The decimal part of the obtained solution from the neural network has been improved by hybrid intelligent algorithm. The obtained outputs from each neural network were evaluated using direct kinematics equations of the robotic manipulator to select the best solution in hybrid intelligent algorithm. Then, the decimal part of this solution was improved up to 10 digits. In the proposed solution system, SA is used as a genetic operator. The best solution of the each population is given to SA as an initial solution. When the SA found the first better solution than the initial, obtained solution is given to GA to generate the new population. Thus, it is provided to decrease as much as possible the number of repetition of the best solution alternative obtained from the genetic algorithm in each stage. Since SA is trying to find a better solution than GA's, the performance of the genetic algorithm will be more increased.

In Fig. 4, the graphical of error change versus number of generations for GA and Hybrid model has been given. In the figure it is seen that the hybrid solution system is reaching the optimal solution faster than genetic algorithm. Thus, the number of generation to reach the optimal solution was reduced.

Hybrid intelligent algorithm has been used as a search algorithm to find the best-fitting 10 digits for the decimal part of the solution. The proposed algorithm has been applied to the four DOF robotic manipulator. The error at the end of learning is reduced, efficiently.

Acknowledgments This work was supported by Research Fund of the Sakarya University. Project Number: 2013-05-02-001.

References

1. Bingul Z, Ertunc HM (2005) Applying neural network to inverse kinematics problem for 6R robot manipulator with offset wrist. In: Proceedings, 7th international conference on adaptive and natural computing algorithms, Coimbra, Portugal, March 2005
2. Çakar T, Köker R, Demir I (2008) Parallel robot scheduling to minimize mean tardiness with precedence constraints using a genetic algorithm. Adv Eng Softw 39(1):47–54
3. Çakar T, Yazgan HR, Köker R (2008) Parallel robot manipulators, new developments. In: Ryu J-H (ed) Parallel robot scheduling with genetic algorithms. I-Tech Education and Publishing, Vienna, pp 153–170
4. Çakar Tarık, Köker R, Yavuz Sarı Y (2012) Parallel robot scheduling to minimize mean tardiness with unequal release date and precedence constraints using a hybrid intelligent system. Int J Adv Robot Syst 2012(9):252. doi:10.5772/54381
5. Çakar T, Köker R, Canay Ö (2015) A new neuro-dominance rule for single-machine tardiness problem with double due date. Neural Comput Appl 26:1439–1450
6. Daunicht W (1991) Approximation of the inverse kinematics of an industrial robot by DEFAnet. In: Proceedings of the IEEE international joint conference on neural networks, Singapore, pp 531–538
7. Deb K, Agrawal S (1999) A niche-penalty method for constraint handling in genetic algorithms. In: Proceedings of the international conference on neural networks and genetic algorithms, Portoroz, Slovenia, 6–9 April, pp 235–243
8. Duffy J (1980) Analysis of mechanisms and robot manipulators. Wiley, New York
9. Featherstone R (1983) Position and velocity transformation between robot end-effector coordinate and joint angle. Int J Robot Res 2(2):33–45
10. Fu KS, Gonzalez RC, Lee CSG (1987) Robotics: control, sensing, vision, and intelligence. McGraw-Hill, New York
11. Hahala J, Fahner G, Echmiller R (1991) Rapid learning of inverse robot kinematics based on connection assignment and topographical encoding (CATE). In: Proceedings of the IEEE international joint conference on neural networks, Singapore, pp 1536–1541
12. Hasan AT, Hamouda AMS, Ismail N, Al-Assadi HMAA (2006) An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 DOF serial robot manipulator. Adv Eng Softw 37:432–438
13. Hasan AT, Ismail N, Hamouda AMS, Aris I, Marhaban MH, Al-Assadi HMAA (2010) Artificial neural network-based kinematics Jacobean solution for serial manipulator passing through singular configurations. Adv Eng Softw 41:359–367
14. Kalra P, Mahapatra PB, Aggarwal DK (2006) An evolutionary approach for solving the multimodal inverse kinematics problem of industrial robots. Mech Mach Theory 41:1213–1229

15. Karlık B, Aydın S (2000) An improved approach to the solution of inverse kinematics problems for robot manipulators. Eng Appl Artif Intell 13:159–164
16. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
17. Köker R (2005) Reliability-based approach to the inverse kinematics solution of robots using Elman's network. Eng Appl Artif Intell 18:685–693
18. Köker R (2011) A neuro-genetic approach to the inverse kinematics solution of robotic manipulators. Sci Res Essays 6(13):2784–2794
19. Köker R (2013) A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. Inf Sci 222:528–543
20. Köker R (2013) A neuro-simulated annealing approach to the inverse kinematics solution of redundant robotic manipulators. Eng Comput 29(4):507–515
21. Köker R, Öz C, Çakar T, Ekiz H (2004) A study of neural network based inverse kinematics solution for a three-joint robot. Robot Auton Syst 49:227–234
22. Korein JU, Balder NI (1982) Techniques for generating the goal-directed motion of articulated structures. IEEE Comput Graph Appl 2(9):71–81
23. Kozakiewicz K, Ogiso T, Miyake N (1991) Partitioned neural network architecture for inverse kinematics calculation of a 6 DOF robot manipulator. In: Proceedings of the IEEE international joint conference on neural networks, Singapore, pp 2001–2006
24. Kucuk S, Bingul Z (2004) The inverse kinematics solutions of industrial robot manipulators. In: Proceedings of the IEEE international conference on mechatronics, Kuşadası, Turkey, pp 274–279
25. Kucuk S, Bingül Z (2014) Inverse kinematics solutions for industrial robot manipulators with offset wrists. Appl Math Model 38:1983–1999
26. Lee GCS (1982) Robot arm kinematics, dynamics, and control. IEEE Comput 15(12):62–79
27. Manocha D, Canny JF (1994) Efficient inverse kinematics for general 6R manipulators. IEEE Trans Robot Autom 10(5):648–657
28. Martin JA, Lope JD, Santos M (2009) A method to learn the inverse kinematics of multi-link robots by evolving neuro-controllers. Neurocomputing 72:2806–2814
29. Martinetz T, Ritter H, Schulten K (1990) Three-dimensional neural net for learning visuomotor coordination of a robot arm. IEEE Trans Neural Netw 1(1):131–136
30. Mayorga RV, Sanongboon P (2005) Inverse kinematics and geometrically bounded singularities prevention of redundant manipulators: an artificial neural network approach. Robot Auton Syst 53:164–176
31. Montano BR, Anandalingam G, Zandi I (2000) A genetic algorithm approach to policy design for consequence minimization. Eur J Oper Res 124:43–54
32. Nearchou AC (1998) Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. Mech Mach Theory 33(3):273–292
33. Oei CK, Goldberg DE, Chang SJ (1991) Tournament selection, niching, and preservation of diversity. Report no. 91011, Urbana IL: University of Illinois at Urbana-Champaign
34. Oyama E, Chong NY, Agah A (2001) Inverse kinematics learning by modular architecture neural networks with performance prediction networks. In: Proceedings of the 2001 IEEE international conference on robotics and automation, Seoul, Korea, 21–26 May
35. Paul RP, Shimano B, Mayer GE (1981) Kinematics control equations for simple manipulators. IEEE Trans Syst Man Cybern SMC-11(6):66–72
36. Pham DT, Castellani M, Fahmy AA (2008) Learning the inverse kinematics of a robot manipulator using the bees algorithm. In: Proceedings of the IEEE international conference on industrial informatics (INDIN 2008) DCC, Daejeon, Korea, 13–16 July 2008
37. Sarı Y (2014) Performance evaluation of the various training algorithms and network topologies in a neural-network-based inverse kinematics solution for robots. Int J Adv Robot Syst 11:64
38. Schlunz EB, Van Vuuren JH (2013) "An investigation into the effectiveness of simulated annealing as a solution approach for the generator maintenance scheduling problem. Electr Power Energy Syst 53:166–174
39. Tabendeh S, Clark C, Melek W (2006) A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering. In: Proceedings of the IEEE congress on evolutionary computation, July 16–21, Vancouver, Canada, pp 1815–1822
40. Tejomurtula S, Kak S (1999) Inverse kinematics in robotics using neural networks. Inf Sci 116:147–164
41. Toshani H, Farrokhi M (2014) Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: a Lyapunov-based approach. Robot Auton Syst 62:766–781
42. Vosniakos GC, Kannas Z (2009) Motion coordination for industrial robotic systems with redundant degrees of freedom. Robot Comput Integr Manuf 25:417–431