A Multi-objective Differential Evolution Algorithm for Robot Inverse Kinematics

Enrique Rodriguez[#1], Baidya Nath Saha[*2], Jesús Romero-Hdz[#3], David Ortega[*4]

[#1]*Universidad Autónoma de Nuevo León, San Nicolás de los Garza, México*
[*2]*Centro de Investigación en Matemáticas(CIMAT), Monterrey, México*
[#3,*4]*Centro de Ingeniería y Desarrollo Industrial(CIDESI), Monterrey, México*

Abstract *— This paper presents the robot inverse kinematics solution for four Degrees of Freedom (DOF) through Differential Evolution (DE) algorithm. DE can handle real numbers (float, double) which leads more powerful than Genetic Algorithm (GA). We propose a multi-objective fitness function that makes an attempt to minimize the positional error and maximum angular displacement of the robot joints. Maximum angular displacement based fitness function adopt the constraints on different unrealistic rotational movement of the manipulator. We employ an equitable treatment of both fitness functions while maximizing these two over generations that iteratively selects the optimal weights of these two fitness functions automatically. Trigonometric mutation and binomial crossover improve the performance of the conventional DE technique. We compared the results of proposed multi-objective DE with GA and Algebraic Method (AM). Proposed multi-objective DE algorithm obtains less positional error than conventional DE, GA and AM while meeting the rotational constraints of the manipulator's joints.*

Keywords— *Inverse Kinematic, Differential Evolution, Multi-objective optimization, Genetic Algorithm, Robot manipulator with four degrees of freedom.*

I. INTRODUCTION

Robot inverse kinematics is a topic largely addressed in robotic research for many years. Advancement of robotics technology are enlarging it areas of application and hence robots are now often used in day-to-day activities of many fields of industry, science, and medicine. This elevates the inverse kinematics problem to the upfront of the robotic research. The inverse kinematics problem is to find the angular position of the robot joints which can achieve some expected position and orientation of the end effector that allows the robot to execute the required task. The angular position of the robot joints is required to transform a motion so that the robot can perform some given tasks such as peg-hole insertion, parts mating and manufacturing assembly operation which are very common in day-to-day industry operation [1].

Robot kinematics problem can be categorized into two classes: forward kinematics problem in which position and orientation of the end effector can be directly computed from the angular position of the robot joints using Denavit-Hartenberg (DH) method and the inverse kinematics problem which is defined above. The inverse kinematics problem is quite complex because it deals with solving a system of underdetermined nonlinear equations. As a result, this is not always possible to find a closed-form solution. Due to the underdetermined nature of the problem, sometimes multiple solutions may exist, however, none of them may not be admissible for the existing kinematic structure of the robot. In some cases, no solution at all may exist, i.e., robot cannot achieve the desired position and orientation of the end effector because it is very difficult to find the suitable constraints for solving the underdetermined system of non-linear equations.

Different solution techniques for this problem can be categorized into two major classes: closed-form analytical and numerical methods. Closed-form solutions are faster than the numerical solutions and it can identify all possible solutions, but these techniques are dependent on robot kinematic structure and it is not possible to obtain for different robot kinematic structures such as Crustcrawler AX-18 Smart Robotic Arm [2]. In contrast, numerical solutions are more general because they are not dependent of the robot structure. However, numerical methods are slower because they normally first guess an initial solution and then find the final solution in an iterative manner and they converge into local optima. The quality of the solution depends on the set of starting values. In addition, when the numerical methods fail to converge, they cannot obtain the solution even if the solution of the inverse kinematics problem exists.

In this research we aim at determining the solution of inverse kinematics problem for Crustcrawler AX-18 Smart Robotic Arm which has four degrees of freedom and a gripper. This kind of robot manipulator are widely used in different industrial applications such as peg-hole insertion tasks, complex manipulations, obstacle avoidance and assistance tasks like serving drink to the users [3]. Though there are few closed form solutions using algebraic method

available in literature [1], [4], [5] for inverse kinematics of Crustcrawler AX-18 robot manipulator, however it does not always guarantee to provide the admissible solutions. To demonstrate this phenomenon, we conducted a simulation experiment using P. Corke's matlab toolbox [6] with a four link robot manipulator with four degrees of freedom. The length of the links are equal to the length of the links of the Crustcrawler AX-18 robot manipulator [2] used in this experiment that are provided in Table 1.

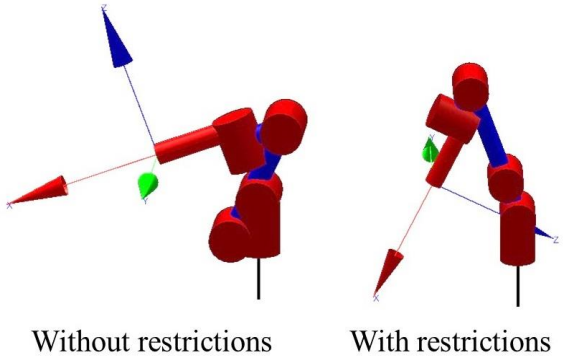

Fig. 1 Two different configurations of the robot achieving the same target position, p_x = 10, p_y =10, p_z =10.

Fig. 1 shows the results of two robot configurations which can achieve the final target location p_x = 10, p_y =10, p_z =10. Fig. 1 is developed using P. Corke's matlab toolbox. We used different evolution algorithm to find the robot inverse kinematics solutions under two different conditions: without restrictions and with restrictions. The restriction includes the real constraint on angular displacement of the servomotor of the Crustcrawler AX-18 robot manipulator based on the capacity of the servomotor [2]. The restrictions are:

$$\theta_1 \in [-150^\circ, 150^\circ],\ \theta_2 \in [-60^\circ, 240^\circ],$$
$$\theta_3 \in [-150^\circ, 150^\circ],\ \theta_4 \in [-150^\circ, 150^\circ].$$

The solutions found without restrictions are θ_1= -135º, θ_2= -123.27º, θ_3= 188.83º, and θ_4= 134.84º and with restrictions are θ_1= 45º, θ_2= 72.42º, θ_3= -2.29º, and θ_4= -133.6º. The results show that the robot configuration without restrictions violate the constraints on θ_2 and θ_3. However, we found a viable solution with restrictions. This experiment shows that multiple solutions exist in robot inverse kinematics problem due to underdetermined nature of the linear systems (we have three linear equations for solving four degrees of freedom of the robot joints). Finding such viable solutions require an exhaustive search which is not always practically feasible for higher dimensions. Existing numerical methods on evolutionary algorithms, to name a few genetic algorithm [7], [8] and differential evolution [9], [10] provide solutions for the problems of exhaustive search with acceptable accuracy. In practice, usually manually designed look-up tables based approximate inverse kinematics based solutions are used for controlling a robotic manipulator [11].

Existing research efforts based on evolutionally algorithms towards robot inverse kinematics mainly deal with minimizing the positional error. However, it is found that even the robot can achieve the target position with minimal position error, the solutions are not admissible solutions (the required robot configuration to achieve the target position is many times practically not feasible) due to the limitations of the servomotor to achieve very high angular displacement as shown in Fig. 1.

However, success of differential evolutions for its faster convergence and more accurate solutions over genetic algorithm (DE can tackle real and floating point numbers which is required for robot joint angles) has attracted to develop a multi-objective differential evolution algorithm for robot inverse kinematics problem. Unlike the existing researches, along with minimizing positional error, we try to minimize the maximum of the angular displacement of robot joints which naturally restrains on robot angular displacement and avoids to find the robot configuration with high angular displacement of the joints and hence assists in finding admissible solutions. Thus the proposed multi-objective differential evolution algorithm offers a practically viable solution for robot inverse kinematics problem through achieving the target position with minimal positional error and satisfying the angular constraints of the robot joints.

This research offers the following technical contributions. Firstly, we propose a multi-objective differential evolution algorithm for robot inverse kinematics problem. Secondly, we proposed two fitness functions: a) the first one minimizes the final positional error of the robot and (b) the second one minimizes the maximum angular displacement of the robot angular joints. The second fitness function restricts the rotational displacement of the angular joints of the robot while the first one attempts to arrive the end effector of the robot to the target position with minimal positional error. Thirdly, we employ an unbiased treatment of both fitness functions that selects the optimal weights between these two functions iteratively over the generations. Fourthly, we exploited binomial crossover and trigonometric mutation for differential evolution approach that accelerates the convergence of the differential evolution algorithm. We implemented the multi-objective differential algorithm on Crustcrawler AX-18 robot manipulator [2] with four degrees of freedom. We also conducted the sensitivity analysis of the proposed algorithm. Experimental results demonstrate that proposed multi-objective differential evolution algorithm achieves less positional error and satisfy the angular constraints of the robot manipulator than genetic algorithm and algebraic method. We also developed the forward kinematics for the Crustcrawler AX-18 robot manipulator using

the Denavit-Hartenberg(DH) method. We modified the inverse kinematics solutions of the algebraic method for Crustcrawler AX-18 robot manipulator.

The organization of the remaining of the paper is as follows. Section II presents the literature review regarding the existing robot inverse kinematics solutions to algebraic method, genetic algorithm and differential evolution. Section III discusses the inverse kinematics problem formulation using algebraic, genetic and DE method. Section IV presents the proposed multi-objective differential evolution for robot inverse kinematics problems. Section V illustrates the experimental results and discussions. Section VI concludes the work.

II. BRIEF LITERATURE REVIEW

In this section, we present the existing solutions of robot inverse kinematics using three algorithms, Algebraic Method (AM), Genetic Algorithm (GA), and Differential Evolution (DE).

A. Algebraic Method (AM)

Sultan and Schwartz [4] presented a solution for the inverse kinematic of a 5 DOF robot arm which is practically a 4 DOF manipulator with a degree of freedom in the gripper. The inverse kinematic was obtained from the transformation matrix and the forward kinematic equations which resulted in two sets of possible solutions depending of the calculation of θ_2 and θ_3. The solution proposed can closely approximate desired points within 1 cm of the workspace boundaries.

Ramirez and Toscano [1] proposed a closed-form solution to the inverse kinematic of a 5 DOF manipulator robot defining the existence conditions for all the possible solutions and the singular configurations were identified. The proposed method uses the desired position of the center of the gripper as well the direction of the gripper's main axis.

Mohammed and Sunar [12] studied the forward and inverse kinematic of a 4 DOF robotic arm. For the forward kinematic model, the problem was compared using the Denavit-Hartenberg convention and the product of exponentials, those two approaches showed an identical solution. In the solution for the inverse kinematic problem an algebraic method was implemented which made use of a fourth parameter besides the *x, y, z* desired point, called the end effector orientation.

B. Genetic Algorithm y(GA)

Joey and David [7] introduced the genetic algorithms for solving the inverse kinematics problem for redundant robots using a single fitness function which integrates the error of the final end effector position and the desired position and an additional term based on the angular joint displacements from the initial position of the robot. The results showed a significantly large final positioning error. They proposed for a future work to employ the Newton-Raphson method to minimize the final error to zero of the genetic algorithm.

F.Y.C. and S.P. [8] used a genetic algorithm to optimize the inverse kinematic for real-time trajectory planning manipulator. Using a new proposed crossover method called Dynamic Multilayered Chromosome Crossover (DMCC) they implemented the method for a planar manipulator of three degrees of freedom. The results indicated an improved of the number of iterations for the genetic algorithm.

Zhen and Yan-Tao [13] proposed a multi population genetic algorithm (MPGA) in order to improve the global converge. Where the MPGA divides the whole population into several populations, then through artificial selection and an immigration operation forms a new population by selecting the best individuals from each category. The MPGA compared with the simple genetic algorithm (SGA) made the global solution more efficient and accelerated the converge speed.

C. Differential Evolution (DE)

Gonzalez and Blanco [9] demonstrated that a memetic approach increases the converge of the differential evolution algorithm for the inverse kinematic problem. They introduced a local search mechanism called discarding. The results using a 3 DOF planar robot showed that the proposed method is able to solve the inverse kinematic accurately and in fewer generations than the conventional DE.

Wang and Hao [10] studied the forward kinematic of a pneumatic parallel manipulator using genetic algorithm, particle swarm optimization and the differential evolution algorithms. The performance of the DE was quite better than the GA and PSO where the speed of the convergence of the DE was less than the other ones with a greater reliability to obtain the global optima for the forward kinematic.

Nguyen and Ho Pham [14] proposed a hybrid differential evolution to train an adaptive MIMO neural network for the solution of inverse kinematic. The hybrid differential evolution algorithm applied to solve the inverse kinematic of a 3 DOF manipulator which is composed by the back-propagation algorithm and the DE algorithm proved a faster performance and better precision than the conventional back-propagation algorithm or the solely differential evolution algorithm.

III. INVERSE KINEMATIC PROBLEM FORMULATION

Given the current (initial) and the desired position of the end effector of the robot that is defined by the user, we formulate the robot inverse kinematic problem is to find the angular position of the end effector to reach the desired position with minimal positional error and the minimum rotational displacement between the current joint angles and the joint angles of the end effector.

A. FORWARD KINEMATICS

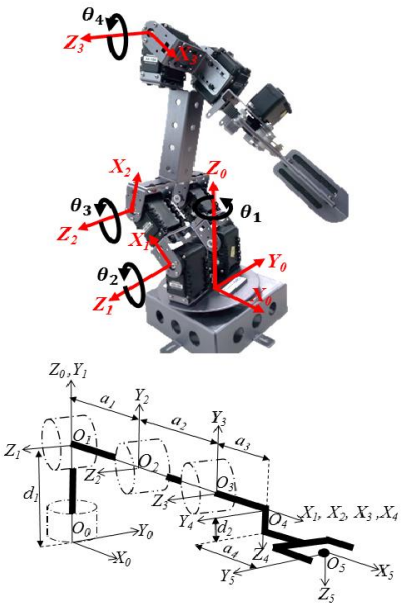

Fig. 2 Four DOF Crustcrawler AX-18 robot manipulator robot configuration

For a kinematic model, a robot manipulator can be considered as a chain of links attached by joints. We can determine the position and orientation of the end effector or TCP (Tool Center Point) given a set of geometrical characteristic of the robot and a base frame, and this analysis is called forward kinematics. We use the Denavit-Hartenberg(DH) convention for the representation of the robot forward kinematic model. The DH convention are described by four parameters (link length, link twist, joint distance and joint angle) as given below [15].

1. Link length (a_i) is the distance between z_{i-1} and z_i axes along the x_i axis, a_i is the kinematic length of link (i).
2. Link twist (α_i) is the required rotation of the z_{i-1} axis about the x_i axis to become parallel to the z_i axis.
3. Joint distance (d_i) is the distance between x_{i-1} and x_i axes along the z_{i-1} axis. Joint distance is also called the link offset.
4. Joint angle (θ_i) is the required rotation of x_{i-1} axis about the z_{i-1} axis to become parallel to the x_i axis.

The robot studied in this paper has four revolute joints, the d_i has a fixed value and θ_i is the variable. After applying the DH convention on this robot manipulator as shown in Figure 1, the values of the four parameters (a_i, α_i, d_i and θ_i) are listed in the Table 1.

Table 1. Denavit-Hartenberg Parameters

Frame	a_i	α_i	d_i	θ_i
O_1	0	$\pi/2$	d_1	θ_1
O_2	a_1	0	0	θ_2
O_3	a_2	0	0	θ_3
O_4	a_3	$\pi/2$	0	θ_4
O_5	a_4	0	d_2	0

For our robot manipulator, the value of d_1= 4.36 cm, a_1 = 6.75 cm, a_2 = 17.23 cm, a_3 = 6.3 cm, d_2 = 2.0 cm and a_4 = 11.7 cm. The DH parameters facilitates to obtain each homogeneous transformation ${}^{i-1}A_i$ for the four coordinate systems of the robot manipulator which is given by as follows.

$$ {}^{i-1}A_i = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

The transformation matrix that links the position and orientation of the end effector or TCP is given by

$$ {}^0T_5 = \prod_{i=0}^{4} {}^iA_{i+1} = \begin{pmatrix} {}^0R_5 & {}^0P_5 \\ 0 & 1 \end{pmatrix} $$

The ${}^0R_5$ matrix is called the rotation matrix that describes the orientation of the frame O_4 relative to the base frame and the vector ${}^0P_5$ is the position of the center of the TCP frame. The matrix for the robot forward kinematics is presented where the notation used are $c_i=cos\theta_i$, $s_i=sin\theta_i$, $c_{ij}=cos(\theta_i+\theta_j)$, $s_{ij}=sin(\theta_i+\theta_j)$, $c_{ijk}=cos(\theta_i+\theta_j+\theta_k)$ and $s_{ijk}=sin(\theta_i+\theta_j+\theta_k)$.

After simplifying the ${}^0P_5$ vector using the product and sum difference identities, we obtain the position of the end effector which is given by,

$$ p_x = a_4c_1c_{234} + d_2c_1s_{234} + a_1c_1c_2 + a_3c_1c_{234} + a_2c_1c_{23} $$
$$ p_y = a_4s_1c_{234} + a_1c_2s_1 + d_2s_1s_{234} + a_3s_1c_{234} + a_2s_1c_{23} $$
$$ p_z = d_1 + a_1s_2 + a_4s_{234} - d_2c_{234} + a_2s_{23} + a_3s_{234} $$

And the end effector orientation as,

$$ \varphi = \theta_2 + \theta_3 + \theta_4 $$

B. APPROACHES TO INVERSE KINEMATIC

We compare three different methods for the inverse kinematics solution, they are described below.

ALGEBRAIC METHOD

To solve the inverse kinematic problem by algebraic method, we use the equations of the forward kinematic and after some algebraic operation using trigonometric identities we have:

$$ \theta_1 = \tan^{-1}(p_y / p_x), $$
$$ \theta_2 = \sin^{-1}(B^2 + C^2 - a_1^2 - a_2^2 \ / \ 2a_1) - \tan^{-1}(B/C) $$
$$ \theta_3 = \cos^{-1}(B^2 + C^2 - a_1^2 - a_2^2 \ / \ 2a_1a_2), $$
$$ \theta_4 = \varphi - \theta_2 - \theta_3, \text{ where} $$
$$ C = p_z - d_1 - a_3\sin\varphi - a_4\sin\varphi + d_2\cos\varphi \text{ and} $$
$$ B = p_y / \sin\theta_1 - a_3\cos\varphi - a_4\cos\varphi - d_2\sin\varphi $$

GENETIC ALGORITHM

Genetics algorithms emulate natural selection of a set of individuals in order to search the best solution for a determined problem. The genetic configuration of each individual is a possible solution. GA starts with an initial population and those are submitted to an evolutionary process in such way that the best adapted individuals will continue to reproduce among them and over several generations the best adapted will stands out. We tailor the genetic algorithm for a multi-objective inverse kinematic solution based on: selection, cross-over and elitism that are discussed below.

String representation of joint angles

The solution for the inverse kinematic implemented with a genetic algorithm starts with encoding the joint angles represented in a binary string of 36 bits long where it is divided in 4 chains of 9 bits for each joint respectively.

Initialization of the population

The initialization of the population is based on binary vectors with a uniform distribution *U (0,1).* Where each vector is called individuals *i=1,2,3...,Np* where *Np* means the size of the population. Their genes are generated randomly.

Selection

When the GA enters to the main loop, the next step is the selection. Using a stochastic method known as roulette wheel selection, it selects the parents form the current population for further imitation of natural selection, where with a better fitness value it is most likely to be selected for breeding. Thus the probability of being selected as one candidate in all the current population is given by [16]:

$$p_i = \frac{f_i}{\sum_{i=1}^{Np} f_i}$$

Where i is the individual in the current population and f_i is its corresponding fitness value.

Decoding the individuals

Each individual has to be decoded in order to using it for the fitness function evaluation. We implement the next equation [8],

$$\theta = \pm\left[\left(\sum_{n=0}^{7} b \bullet 2^n\right)\frac{180}{255}\right]$$

Where θ correspond for each joint angle for a set of 9 bits. The first bit from the set represented by the mathematical sign (±) in the equation determines the rotating direction of the joint angle, 1 for positive and 0 for a negative rotation. b is the bit that can be either 0 or 1 dynamically.

Fitness Function

For this paper we have implemented a multi-objective fitness function that takes into account the error of the difference of the target position and the proposed manipulator end effector point and the maximum angular displacement between the final joint angles and the initial joint angle given by the user.

f_1 is a sub-function that determines the difference between the target position and the end effector position and it is computed by the forward kinematic equation. The equation of the fitness function is defined by,

$$f_1(P,P_e) = 1 - \exp\left(-\sqrt{(p_x - p_{x_e})^2 + (p_y - p_{y_e})^2 + (p_z - p_{z_e})^2}\right)$$

Where $P = \{p_x, p_y, p_z\}$ and $P_e = \{p_{x_e}, p_{y_e}, p_{z_e}\}$ are the final target and the end effector position respectively. f_2 is the second sub-function that takes into account the maximum rotational displacement and it is computed by the following equation,

$$f_2(\Theta_I, \Theta_{e,g,i}) = \max\{|\theta_{1I}, \theta_{1e,g,i}|, |\theta_{2I}, \theta_{2e,g,i}|, |\theta_{3I}, \theta_{3e,g,i}|, |\theta_{4I}, \theta_{4e,g,i}|\}$$

Where $|\theta_{kI}, \theta_{ke,g,i}| = |\theta_{kI} - \theta_{ke,g,i}|$ for $k = \{1,2,3,4\}$ and $\Theta_I = \{\theta_{1I}, \theta_{2I}, \theta_{3I}, \theta_{4I}\}$,

$\Theta_{e,g,i} = \{\theta_{1e,g,i}, \theta_{2e,g,i}, \theta_{3e,g,i}, \theta_{4e,g,i}\}$

are the initial angular position vector and final angular position vector for individual i after generation g for the robot respectively where each element of the vector represents the angle of each joint.

Converge analysis of genetic algorithm

Aytug and Koehler [17], [18] showed that for a general Markov Chain model of genetic algorithm with elitism, an upper bound for the number of iterations t required to generate a population $S+$ which consists entirely of minimal solutions has been generated with probability $\alpha \in (0,1)$; is given by,

$$t \geq \left| \ln(1-\alpha) / n \ln\left(1 - \min\left\{\mu^l, (1-\mu)^l\right\}\right)\right|$$

Where, l is the length of the chains that represent the individual, n is the population size and $\mu \in (0,1)$ is the mutation rate. $\lceil x \rceil$ is the smallest integer greater than or equal to x. Studniarski [19] showed that for multi-objective optimization, the (possibly unknown) number m of these solutions is bounded from below by some known positive integer $\overline{m}$. Suppose also that there exists a number $\beta \in (0,1/\overline{m})$,an upper bound for the number of iterations t is given by,

$$t \geq \left| \ln(1-\alpha) / \ln\left(1 - (\overline{m}\beta)^l\right)\right|$$

If no non-trivial lower bound $\overline{m}$ is known, we may always use $\overline{m} = 1$.

DIFFERENTIAL EVOLUTION (DE)

The main difference between *DE* and other Evolutionary Algorithms (EAs) is the implementation of the mutation operation. The mutation operation of DE applies the vector differentials between the existing population members for determining both degree and direction of the operation of the perturbation applied to the individual subject of the mutation operation.

Initialization

The algorithm starts with a dynamic initial population $P = \{i_j\}$, where the elements of the population are called "individuals*" i = ,2,3,…,j*. Their genes are generated randomly.

For the inverse kinematic problem with a four dimensional vector and *Np*, the size of the population, the DE algorithm has a population of size *Np* joint configuration,

$$P_g = \{\vec{q}_{1,g}, \vec{q}_{2,g}, ..., \vec{q}_{Np,g}\}$$

Where g is the number of generation, *g=0,1,…,gmax* and $\vec{q}$ is a vector array also called as individual codded as a floating point of *D=4* length, for the solution of the inverse kinematic of 4 DOF each individual is a four dimensional vector defined as:

$$\vec{q}_{i,g} = [q_{1,i,g}, q_{2,i,g}, q_{3i,g}, q_{4,i,g}] = [\theta_{1,i,g}, \theta_{2,i,g}, \theta_{3,i,g}, \theta_{4,i,g}]$$

For all *i=1, 2,…,Np*. At the generation *g=0* the DE starts with an initial population generated by a randomly uniform distribution with a search space defined by the joint upper and lower bounds as

$$\vec{q}_{i,0} = \vec{q}^{Lo} + rand(0,1)\left(\vec{q}^{Hi} - \vec{q}^{Lo}\right)$$

For the robot configuration presented in this paper we consider the joints limits

$$\vec{q}^{Lo} = \left[\frac{-5\pi}{6}, \frac{-\pi}{3}, \frac{-5\pi}{6}, \frac{-5\pi}{6}\right]$$

$$\vec{q}^{Hi} = \left[\frac{5\pi}{6}, \frac{4\pi}{3}, \frac{5\pi}{6}, \frac{5\pi}{6}\right]$$

Mutation

In each generation the mutation process begins with the selection of three randomly individuals of the population. The $\vec{q}_{i,g}$ individual to be perturbed is called the target vector which could be replaced by a mutant vector also known as donor vector. The $\vec{v}_{i,g}$ donor vector is obtained through the differential mutation operation based on three chosen individuals. The difference of any two of these three vectors is scaled by a scalar factor *F* and the scaled difference is added to the remaining vector to obtain the donor vector which is given by,

$$\vec{v}_{i,g} = \vec{q}_{r_1^i,g} + F \cdot (\vec{q}_{r_2^i,g} - \vec{q}_{r_3^i,g})$$

Where from *i=1,2,…,Np* on the generation g for each target vector, the *i*th difference vectors $\vec{q}_{r_2^i g}$ and $\vec{q}_{r_3^i g}$, and the base vector $\vec{q}_{r_1^i,g}$ belong to the current population P_g. The indices r_{1g}^i, r_{2g}^i and r_{3g}^i are randomly chosen from $i = [1,..,Np]$ in such a way that they are mutually exclusive.

Beside the mutation presented before we implemented the trigonometric mutation [20], when the trigonometric mutation operation is performed, instead of an individual randomly taken from the three chosen ones as the original mutation of DE, the donor to be perturbed is taken to be the center point of the hypergeometric triangle. The mutation operation is performed according to the following equations:

$$\vec{v}_{i,g} = (\vec{q}_{r_1^i,g} + \vec{q}_{r_2^i,g} + \vec{q}_{r_3^i,g})/3 + (p_2 - p_1)(\vec{q}_{r_1^i,g} - \vec{q}_{r_2^i,g}) + (p_3 - p_2)(\vec{q}_{r_2^i,g} - \vec{q}_{r_3^i,g}) + (p_1 - p_3)(\vec{q}_{r_3^i,g} - \vec{q}_{r_1^i,g})$$

Where

$$p_1 = |f(\vec{q}_{r_1^i,g})|/p',\ p_2 = |f(\vec{q}_{r_2^i,g})|/p',\ p_3 = |f(\vec{q}_{r_3^i,g})|/p',$$

$$p' = |f(\vec{q}_{r_1^i,g})| + |f(\vec{q}_{r_2^i,g})| + |f(\vec{q}_{r_3^i,g})|$$

Crossover

For the purpose to increment the diversity and as well the enrichment of the mutation strategy used in the step before, we used the binomial crossover. The crossover step takes the donor vector $\vec{v}_{i,g}$ to exchanges its components with the target vector $\vec{q}_{i,g}$ that is regulated by a constant crossover rate $Cr \in [0,1]$. As result we form the trial vector $\vec{u}_{i,g} = \{u_{1,i,g}, u_{2,i,g}, ..., u_{D,i,g}\}$ as follows:

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & if\ (rand_{j,i}(0,1) \le Cr) \\ q_{j,i,g} & otherwise \end{cases}$$

Where *i=1,2,…,Np* and *j=1,2,…,D* and $rand_{j,i}(0,1)$ is a uniformly distributed random number.

Selection

After obtaining the fitness function of the trial vector and the target vector we compare them through the Greedy selection which one is better. If the trial vector is better than the target vector, the trial vector will replace it for the next generation otherwise the target vector will stay as it is presented in the equation

$$\vec{q}_{i,g+1} = \begin{cases} \vec{u}_{i,g} & if\quad f(\vec{u}_{i,g}) \ge f(\vec{q}_{i,g}) \\ \vec{q}_{i,g} & otherwise \end{cases}$$

Where f is the maximized fitness function used for the differential evolution algorithm.

Pseudocode 1. Multi-objective Differential Evolution

Function θ_{final} = DE_Robot_kinematics (P, Θ_I)

% input P represents coordinates of target point

% $\Theta_I = \{\theta_{1I}, \theta_{2I}, \theta_{3I}, \theta_{4I}\}$, Θ_I is the initial angular

% position vector.

% And output θ_{final} represents the corresponding angle of the robot joints

1. % Parameters initialization

$g \leftarrow 0$, $g_{max} \leftarrow 200$, $M \leftarrow 0.2$, Cr$\leftarrow 0.8$, $F \leftarrow 0.2$, $Np \leftarrow 150$

2. % Initial population with respect to limits

Pop $\leftarrow$ $\Theta_{e,g,i}$; $\Theta_{e,g,i}$ = [$\theta_{1e,g,i}$, $\theta_{2e,g,i}$, $\theta_{3e,g,i}$, $\theta_{4e,g,i}$],

i = 1,2, .., Np

f_1 ($\Theta_{e,g,i}$) = exp(-err); err = ||P -fwdkine($\Theta_{e,g,i}$)||

f_2 ($\Theta_{e,g,i}$) = exp(-angdist); angdist = max(Θ_I , $\Theta_{e,g,i}$)

$f_2(\Theta_I, \Theta_{e,g,i}) = \max\{|\theta_{1I}, \theta_{1e,g,i}|, |\theta_{2I}, \theta_{2e,g,i}|, |\theta_{3I}, \theta_{3e,g,i}|, |\theta_{4I}, \theta_{4e,g,i}|\}$

$f = w_1 f_1 + w_2 f_2$, $w_1 = 0.5$, $w_2 = 0.5$

3. **while** (g <= g_{max})

4. **for** (i = 0; i <= N_p; i++)

5. $r_1 \leftarrow$rand(N_p), $r_2 \leftarrow$rand(N_p), $r_3 \leftarrow$rand(N_p)

6. % Mutation

m$\leftarrow$ rand(0,1)

if m< P_m **{**

if m<=Mt

Compute u_i using trigonometric mutation

else

7. $u_i \leftarrow x_{r3} + F(x_{r1} - x_{r2})$

end if

8. **else**

$u_i \leftarrow$ $\Theta_{e,g,i}$

end if

9. % Crossover

if (rand(0,1) < P_c)

10. $y_i \leftarrow u_i$**}**

11. **else**

$y_i \leftarrow$ $\Theta_{e,g,i}$

end if

12. %Selection

if $f(y_i) > f(\Theta_{e,g,i})$

$\Theta_{e,g,i} \leftarrow y_i$

13. **end if**

end for

14. g$\leftarrow$g+1

end while

$\Theta_{final} \leftarrow \arg\max_{\Theta_{e,g_{\max},i}} f(\Theta = \Theta_{e,g_{\max},i})$

IV. MULTI-OBJECTIVE DIFFERENTIAL EVOLUTION FOR INVERSE KINEMATICS SOLUTIONS

In this paper we have implemented a multi-objective fitness function that takes into account the positional error and maximum angular displacement of the robot joints and returns a real number (weighted linear combination of positional error and maximum angular displacement of the robot joints), $f(\vec{q}_{i,g}) \Rightarrow \Re$ that measures the adaptability of each sequence.

$$f(\vec{q}_{i,g}) = \sum_{h=1}^{H} w_h f_h$$

Where w_h is the weight that defines the importance of each sub-function. These weights are computed dynamically in each iteration g with the equation [21],

$$w_{\tilde{h}}(g) = \frac{\sum_{h=1, ih \neq \tilde{h}}^{H} \left| f_h(\vec{q}_{b,g-1}) \right|}{(H-1) \times \sum_{h=1}^{H} \left| f_h(\vec{q}_{b,g-1}) \right|}$$

Where H is the number of sub-functions, h = 1…H, $\vec{q}_{b,g-1}$ is the best individual among solutions of the population in the previous generation P_{g-1}. $w_{\tilde{h}}(g)$ is the dynamic weight satisfying the following conditions,

$$0 \leq w_{\tilde{h}}(g) \leq 1 \text{ and } \sum_{h=1}^{H} w_{\tilde{h}}(g) = 1$$

Where g represents the iteration step of the GA algorithm. f_1 is a sub-function that takes into account the difference between the target position and the final position of the end effector and it is computed by forward kinematics algorithm presented in section IIIA. The equation of the fitness function is denied by

$$f_1(P, P_e) = 1 - \exp(-\sqrt{(p_x - p_{x_e})^2 + (p_y - p_{y_e})^2 + (p_z - p_{z_e})^2})$$

Where $P = \{p_x, p_y, p_z\}$ and $P_e = \{p_{x_e}, p_{y_e}, p_{z_e}\}$ are the final target and the end effector position respectively. f_2 is the second sub-function that takes into account the maximum rotational displacement and it is computed by the following equation [7]

$$f_2(\Theta_I, \Theta_e) = \max\{|\theta_{1I}, \theta_{1e}|, |\theta_{2I}, \theta_{2e}|, |\theta_{3I}, \theta_{3e}|, |\theta_{4I}, \theta_{4e}|\}$$

Where $|\theta_{kI}, \theta_{Ki}| = |\theta_{kI} - \theta_{Ki}|$ for $k = \{1,2,3,4\}$ and $\Theta_I = \{\theta_{1I}, \theta_{2I}, \theta_{3I}, \theta_{4I}\}$, $\Theta_i = \{\theta_{1i}, \theta_{2i}, \theta_{3i}, \theta_{4i}\}$ are the initial and final rotational displacement vector after iteration g for the robot respectively where each element of the vector represents the angle of each joint.

$\theta_{1i} \leftarrow$ rand (-5π/6, 5π/6), $\theta_{2i} \leftarrow$ rand (-π/3, 4π/3), $\theta_{3i} \leftarrow$ rand (-5π/6, 5π/6), $\theta_{4i} \leftarrow$ rand (-5π/6, 5π/6).

We computed the crossover probability as follows [22],

$$P_c = \begin{cases} \dfrac{Cr(f_{\max} - f)}{(f_{\max} - f_{avg})} & f \geq f_{avg} \\ Cr & f < f_{avg} \end{cases}$$

As well the mutation probability P_m is defined as shown below [22],

$$P_m = \begin{cases} \dfrac{M(f_{\max} - f)}{(f_{\max} - f_{avg})} & f \geq f_{avg} \\ M & f < f_{avg} \end{cases}$$

Determination of population size

The minimum population size *Np* required for maintaining *c* species of equal fitness for *g* generations with probability *g* is determined by the equation:

$$n = \ln((1-\gamma^{\frac{1}{G}})/c)/\ln((c-1)/c)$$

For *c* = *10* optima, generations *G*=100 with probability *g* = 0.999, the required population size, *n* = 131 individuals, so a population of 200 is enough to maintain subpopulations.

The pseudocode of the multi-objective Differential Evolution (DE) is given in pseudocode 1.

V. . EXPERIMENTAL RESULTS AND DISCUSSIONS

We conducted the experiment on Crustcrawler AX-18 robot manipulator with four degrees of freedom as illustrated in Fig. 2. The length of the links is provided in Table 1. We run the algorithm for 200 number of generations and we take 200 samples for each generation. Table 2 demonstrates the positional errors of GA, AM and proposed multi-objective DE algorithm. Table 2 illustrates that multi-objective DE offers less positional errors for all the five randomly selected target positions than GA and AM method. For unbiased comparison, we select the value of φ automatically for AM method which provides the minimum positional error. We run GA algorithm for 400 generations and each generation we allow the best 100 genes.

Table 2. Target positions and the corresponding final positions of the end effector of the robots

Target Position	End Effector Position		
	GA	AM	Proposed DE
(10,10, 10)	(10.8,8.8,11.9)	(10.1,10.1,10)	(10,10,10)
(0, 7.5, 12)	(2.7,7.4,11.3)	(0, 6.8, 11.3)	(0,7.5,12)
(2, 13, -5)	(2.9,15.4,-6.1)	(2.1, 13.5, -5.8)	(2, 13 -5)
(-6, 9.5, 4.3)	(-9.5,9.0,3.9)	(-5.9, 9.3, 4.3)	(-6, 9.5, 4.3)
(3, -7, 13)	(2.0,-9.5,16.1)	(3, -7, 13)	(3, -7, 13)

Fig. 3 shows the mean diversity of the population at each generation for multi-objective differential evolution with three different mutation and crossover configurations: (a) fixed crossover and normal mutation method; (b) binomial crossover but without trigonometric mutation; and (c) binomial crossover and trigonometric mutation.

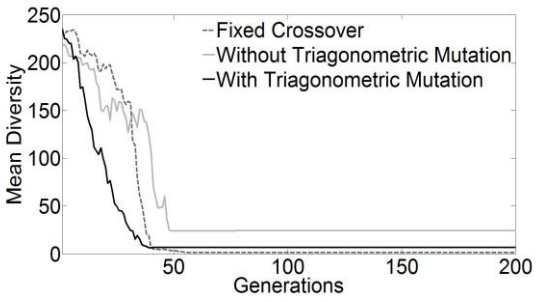

Fig. 3 Mean diversity of the population at each generation for multi-objective DE with three mutation and crossover configurations.

Fig. 3 shows that the mean diversity of the population for binomial crossover and trigonometric mutation decreases faster than normal crossover and mutation method which illustrates faster convergence of the binomial crossover with trigonometric mutation

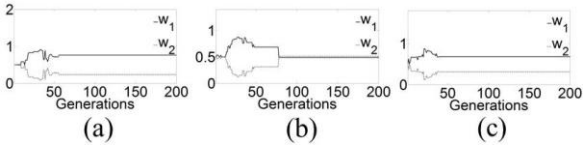

Fig. 4 weights (w_1 and w_2) of two fitness functions for multi-objective DE over generations at three mutation and crossover configurations: (a) fixed crossover, (b) without trigonometric mutation and (c) with trigonometric mutation.

Fig. 4 illustrates weights (w_1 and w_2) of two fitness functions over generations for multi-objective DE for the above three conditions. Fig. 4 demonstrates that binomial crossover with trigonometric mutation (c) selects the optimal weight values of w_1 and w_2 much faster (less than 50 generations) than fixed crossover and without trigonometric mutation (option (a) and (b)). This also illustrates the faster convergence of binomial crossover and trigonometric mutation than fixed crossover and mutation technique. The similar results were found in [20].

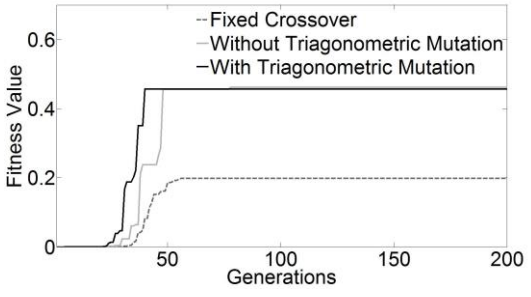

Fig. 5 Fitness value of the best individual at each generation for multi-objective DE with three mutation and crossover configurations.

Fig. 5 illustrates the fitness values of the best individuals over generations Fig. 5 illustrates the fitness values over generations for the three different crossover and mutation methods. Fig. 6 (b) illustrates

that binomial crossover with trigonometric mutation generates lower positional error than fixed crossover and without trigonometric mutation.

Fig. 6 (a) illustrates the pareto front. Results of the pareto front shown in Fig. 6 (a) demonstrate that both positional error and maximum angular displacement cannot be simultaneously reduced, decreasing the value of one increases the value of the other and vice versa. In this study, utilizing iterative dynamic weight selection based multi-objective DE algorithm, we find the optimal weight between positional error and maximum angular displacement of robot joints through fair treatment of both of the fitness functions.

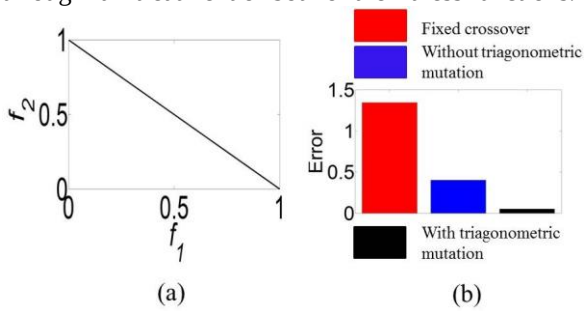

Fig. 6 (a) Pareto front for multi-objective DE; (b) Positional error multi-objective DE with three mutation and crossover configurations.

VI. CONCLUSIONS

This paper presents a multi-objective fitness function for Differential Evolution (DE) algorithm for robot inverse kinematics problem. The values of the two fitness functions are minimized in an iterative fashion over generations. We adopt an equitable treatment that offers optimal weights between these two fitness functions where maximizing the values of both fitness functions over generations. The first fitness function attempts to minimize the positional error while the other fitness function takes into account of maximum angular displacement of the robot joints that attempts to satisfy the constraints on angular displacement of robot joints. We also exploited the trigonometric mutation and binomial crossover that enhances the performance of conventional mutation and crossover method. Binomial crossover expedites the convergence of DE algorithm. We implemented the proposed multi-objective differential evolution algorithm on Crustcrawler AX-18 robot manipulator. We performed the same robot inverse kinematics experiment using Genetic Algorithm (GA) and Algebraic Method (AM). Experimental results demonstrate that Multi-objective DE obtains less positional error and maximum angular displacement than GA and AM method.

ACKNOWLEDGMENT

We would like to acknowledge CIDESI and CIMAT to conduct the experiment reported in this paper and Conacyt to provide associated research support.

REFERENCES

[1] J. G. Ramírez-Torres, G. Toscano-Pulido, A. Ramírez-Saldívar, and A. Hernández-Ramírez, "A complete closed-form solution to the inverse kinematics problem for the P2Arm manipulator robot," *Proc. - 2010 IEEE Electron. Robot. Automot. Mech. Conf. CERMA 2010*, pp. 372–377, 2010.

[2] M. A. Mikulski and T. Szkodny, "Remote control and monitoring of AX-12 robotic arm based on windows communication foundation," *Adv. Intell. Soft Comput.*, vol. 103, pp. 77–83, 2011.

[3] E. P. Lana, B. V Adorno, and C. J. Tierra-Criollo, "Assistance Task Using a Manipulator Robot and User Kinematics Feedback," *XI Simpósio Bras. Automação Intel.*, pp. 1–6, 2013.

[4] H. Sultan and E. M. Schwartz, "Robotic Arm Manipulator Control for SG5-UT," vol. 00, no. 407, pp. 1–5, 2007.

[5] J. Q. Gan, E. Oyama, E. M. Rosales, and H. Hu, "A complete analytical solution to the inverse kinematics of the Pioneer 2 robotic arm," *Robotica*, vol. 23, no. 1, pp. 123–129, 2005.

[6] P. Corke, *Robotics, Vision and Control - Fundamental Algorithms in MATLAB*. 2011.

[7] J. K. Parker, a. R. Khoogar, and D. E. Goldberg, "Inverse Kinematics of Redundant Robots Using Genetic Algorithms," *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 271–276, 1989.

[8] F. Y. C. Albert, S. P. Koh, S. K. Tiong, C. P. Chen, and F. W. Yap, "Inverse kinematic solution in handling 3R manipulator via real-time genetic algorithm," *Proc. - Int. Symp. Inf. Technol. 2008, ITSim*, vol. 4, 2008.

[9] C. González, D. Blanco, and L. Moreno, "A memetic approach to the inverse kinematics problem," *2012 IEEE Int. Conf. Mechatronics Autom. ICMA 2012*, pp. 180–185, 2012.

[10] X. S. Wang, M. L. Hao, and Y. H. Cheng, "On the use of differential evolution for forward kinematics of parallel manipulators," *Appl. Math. Comput.*, vol. 205, no. 2, pp. 760–769, 2008.

[11] A. Henning, "Approximate Inverse Kinematics Using a Database," 2014.

[12] A. A. Mohammed and M. Sunar, "Kinematics Modeling of a 4-DOF Robotic Arm," pp. 87–91, 2015.

[13] Z. Sui, L. Jiang, Y. Tian, and W. Jiang, "Proceedings of the 2015 Chinese Intelligent Automation Conference," vol. 338, no. 5988, pp. 151–161, 2015.

[14] N. N. Son, H. P. H. Anh, and T. Dinh Chau, "Inverse kinematics solution for robot manipulator based on adaptive MIMO neural network model optimized by hybrid differential evolution algorithm," *2014 IEEE Int. Conf. Robot. Biomimetics, IEEE ROBIO 2014*, pp. 2019–2024, 2014.

[15] R. N. Jazar, *Theory of Applied Robotics: Kinematics, Dynamics, and Control*. 2010.

[16] Y. X. and M. Gen, *Introduction to evolutionary algorithms*. Springer-Verlag London, 2010.

[17] H. Aytug and G. J. Koehler, "Stopping criteria for finite length genetic algorithms," *INFORMS J. Comput.*, vol. 8, no. 2, pp. 183–191, 1996.

[18] H. Aytug and G. J. Koehler, "New stopping criterion for genetic algorithms," *Eur. J. Oper. Res.*, vol. 126, no. 3, pp. 662–674, 2000.

[19] M. Studniarski, "Stopping criteria for genetic algorithms with application to multiobjective optimization," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6238 LNCS, no. PART 1, pp. 697–706, 2010.

[20] H. Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *J. Glob. Optim.*, vol. 27, no. 1, pp. 105–129, 2003.

[21] M. Gabli, E. M. Jaara, and E. B. Mermri, "A Genetic Algorithm Approach for an Equitable Treatment of Objective Functions in Multi-objective Optimization Problems," no. May, 2014.

[22] Y. D. Zhao and X. X. Qiao, "Research on optimal multiple sequence alignment," *Proc. Int. Conf. E-bus. E-Government, ICEE 2010*, pp. 5500–5505, 2010.