A New Heuristic Approach for Inverse Kinematics of Robot Arms

Tuğrul Çavdar$^{1,*}$, M. Mohammad R. Alavi Milani$^2$

$^1$Department of Computer Engineering, Karadeniz Technical University, Trabzon, Turkey

Inverse kinematics of a robot arm has become very important research area in recent decades. Also, the use of bio-inspired algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Harmony Search (HS) has been expeditiously increasing to handle many optimization problems. In this paper, a new approach based on the modified Artificial Bee Colony (ABC) has been proposed to solve inverse kinematics problem of robot arms. For this purpose, we defined a cost function based on the Euclidian distance from a position to the target in Cartesian space. We obtained a fitness function from the cost function, and used it in the modified ABC to minimize the fitness value. Also we used some statistical analysis methods to determine the trade-off parameters of the modified ABC. The simulation results show that proposed approach has better performance in terms of both position accuracy and the solution time than the previous studies employing the other heuristic methods like PSO and HS.

**Keywords:** Inverse Kinematics, Robotics, Artificial Bee Colony, Particle Swarm Optimization, Harmony Search, PUMA

1. INTRODUCTION

There have been many topics related to robotics, such as motion planning, intelligent control and tracking control. The changes in joint space generate a motion in Cartesian space. The set of joint variables should be determined to obtain a desired position of the end-effector in Cartesian coordinate. Thus, there is a need to translate from Cartesian space to joint space, which is known as inverse kinematics, while forward kinematics deals with transformation from joint space to Cartesian space. Many algebraic and numerical methods have been developed for inverse kinematics. Inverse kinematics problem is suitable for the heuristic methods because it has multiple solutions. These methods produce better solutions even for the NP problems in a reasonable time. In the recent years, bio-inspired algorithms have been applied for inverse kinematics problem for five or more degree-of-freedom (DOF) robot arms because of offering more accurate results than the traditional methods. Neural Networks, Genetic Algorithm (GA), Harmony Search (HS) and Particle Swarm Optimization (PSO) algorithms have been used to solve the inverse kinematics problem of many robot systems.

In this study, a novel approach based on the modified Artificial Bee Colony (ABC) optimization is developed to solve the inverse kinematics problem, and it has been applied to 6-DOF PUMA 560 robot arm. The proposed approach offers minimum position error in minimum time than the previous studies. In Section 2, the system model of PUMA 560 robot arm is described. In Section 3, the modified Artificial Bee Colony method is discussed. Section 4 presents the implementation of inverse kinematics problem into the modified ABC. The proposed approach is simulated in Section 5. Section 6 concludes.

2. SYSTEM MODEL OF PUMA 560

The PUMA 560 robot has 6-DOF with six revolute joints. Whilst the first three joints locate the end-effector, the following three joints provide the orientation of the end-effector. The PUMA 560 manipulator is shown in figure 1.

---

$^*$ Email Address: ulduz@ktu.edu.tr
Denavit-Hartenberg (D-H) parameters of PUMA 560 are given in Table 1. Eq. 1 gives the multiplication of the transformations of all neighboring frames. This equation will be used to calculate the fitness function of the proposed approach.

\[
0T_6=0T_1^T0T_2^T0T_3^T0T_4^T0T_5^T0T_6
\]  

where \(0T_i\) is the transfer matrix of link \(i\). \(0T_6\) matrix produces a Cartesian coordinate for any six joint angles. Because the cost function of the proposed approach is the Euclidian distance in Cartesian space between the obtained and the target points, \(0T_6\) can be used to calculate the Cartesian coordinate of the obtained point in the cost function. The result of Eq. 1 is a 4×4 transformation matrix given in Eq. 2.

\[
0T_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\]  

where

\[
\begin{align*}
 r_{11} &= c_1[c_23 (c_24 c_5 c_6+c_5 s_6)-s_23 s_5 c_6]+s_1(s_23 c_5 c_6+c_5 s_6), \\
r_{12} &= s_1[c_23 (c_24 c_5 c_6+c_5 s_6)-s_23 s_5 c_6]-c_1(s_23 c_5 c_6+c_5 s_6), \\
r_{13} &= -s_23 (c_24 c_5 c_6+s_5 s_6)+c_23 s_5 c_6, \\
r_{21} &= c_1[c_23 (c_24 c_5 s_6+s_5 c_6)+s_23 s_5 s_6]+s_1(c_24 c_5 s_6+c_5 s_6), \\
r_{22} &= s_1[c_23 (c_24 c_5 s_6+s_5 c_6)+s_23 s_5 s_6]-c_1(c_24 c_5 s_6+c_5 s_6), \\
r_{23} &= -s_23 (c_24 c_5 s_6-s_5 s_6)+c_23 s_5 s_6, \\
r_{31} &= -c_1(c_23 c_5 c_5+s_5 c_6)-s_1 s_5 s_6, \\
r_{32} &= s_1(c_23 c_5 c_6+s_5 s_6)+c_23 s_5 s_6, \\
r_{33} &= c_23 c_5 c_6+s_5 c_5, \\
p_x &= c_1[a_2 c_2+a_3 c_{23}-d_4 s_{23}]-d_3 c_1, \\
p_y &= a_3 s_{23}-a_2 s_2-d_4 c_{23}, \\
p_z &= s_1[a_2 c_2+a_3 c_{23}-d_4 s_{23}]+d_3 c_1.
\end{align*}
\]

3. THE MODIFIED ARTIFICIAL BEE COLONY ALGORITHM

Heuristic Algorithms such as Ant Colony, Harmony Search and Particle Swarm Optimization have been developed to solve optimization problems and even NP problems. Recently, bio-inspired algorithms have been very popular among heuristic methods. One of these methods, called Artificial Bee Colony (ABC) based on search of food by honey bees, was presented by Karaboğ in 2005. In the ABC algorithm, bees are divided into three groups: employed bees, onlooker bees and scouts. In ABC, one employed bee is chosen for each food source. That means the number of employed bee is equal to the number of food source. Search of food source is the duty of employed bees. Foods are brought by employed bees to the hive. Employed bees start to dance in the dance area of hive. Onlooker bees choose one of the food sources according to the dance of employed bees. The number of onlooker bees usually equals to the number of employed bees. The standard ABC method was created based on the above concepts. The brief explanation of the standard ABC by Karaboğ is given in the following subsections.

3.1. INITIAL FOOD SOURCES

In this phase, algorithm generates food source from the search space. Initial food sources are produced randomly within the range of the dimension boundaries of the search space.

The number of food source (SN) is determined according to problem, and the number of dimension (D) in search space is equal to the number of optimization parameters. Also, a counter, assigned for each food source, stores the numbers of trial solution, and this counter is reset initially. In the end of this phase, the food sources are generated to be used by employed, onlooker and scout bees in the next phases.

3.2. EMPLOYED PHASE

In this phase, each employed bee modifies its food source (m) and generates an optimum neighbor food source (v). To generate v, first k and j integer index numbers are randomly chosen as \(j = 1...D, k = 1...SN\). The neighborhood between \(j^{th}\) parameter of \(m\) and \(j^{th}\) parameter of \(k\) is given in Eq. 4.

\[
v_{ij} = m_{ij} + Rand (-1,1)(m_{ij} - m_{kj}) \]  

where \(k\) should be different than \(i\). Then employed bee evaluates quality of \(v\). If the evaluation shows improvement, \(v\) is replaced with \(m\). If \(v_{ij}\) exceeds the boundary of \(j^{th}\) dimension, \(v_{ij}\) is set to a valid value inside the boundary.
After every change in any parameters of food sources, a fitness value is calculated by Eq. 5.
\[
\text{fitness}_i = \begin{cases} 
\frac{1}{1+f_i} & f_i \geq 0 \\
1+\text{abs}(f_i) & f_i < 0 
\end{cases} 
\] (5)
where \(f_i\) is the cost function of \(i^{th}\) food source. In every cycle of this phase, the employed bee memorizes all optimum food sources and forgets the rest. Then the bee resets counter if the new food source is better than the old one, otherwise, only increases counter by 1.

### 3.3. Probability of Food Sources

In this phase, employed bees share their information about food sources with onlooker bees in the dance area. In the standard ABC, onlooker bees choose their food sources in compliance with the probability \(p\) of food source. This probabilistic selection is done with roulette wheel scheme. The probability of \(i^{th}\) food source:
\[
\begin{aligned}
p_i = & \frac{\text{fitness}_i}{\sum_{i=1}^{N}\text{fitness}_i} \\
\end{aligned}
\] (6)
The more probability of food source is, the more onlooker bee visits in roulette wheel scheme.

### 3.4. Onlooker Phase

In this phase, a random number between 0 and 1 is generated for each food source. If generated random number is less than the probability value of the associated food source, this food source is assigned to an onlooker bee. Then the bee modifies this food source in the same method as in Eq. 4, and update the counter.

### 3.5. Scout Phase

The counter is continuously updated and checked by scouts after every cycle in the phases mentioned in Sections 3.2 and 3.4. If the counter of food source is greater than the control parameter named “limit” of the ABC algorithm, the food source is replaced with a new food source via scout with the same method in Section 3.1. In standard ABC, it is assumed that only one food source can be replaced in each cycle. If more than one counter exceeds the “limit” value, maximum one is selected.

### 3.6. Modified Artificial Bee Colony

To improve the search abilities of ABC, many researchers have developed different methods. Tsai et al.\(^{21}\) proposed an enhanced ABC algorithms (IABC). Zhu and Kwong\(^{22}\) presented a novel improved ABC variant. Additionally Karaboğa proposed a modified ABC algorithm\(^{23}\) that we use in this paper. The standard ABC is poorer in convergence rate and the modified version tries to improve this rate.

The modified ABC offers two main differences than the standard ABC: First, the modified ABC may change more than one parameter of food source in every production of neighborhood while the standard ABC changes only \(j^{th}\) parameter. When a food source needs to be modified, a real random number \(R_j\) between 0 and 1 is generated for each parameter of the food source. All parameters whose \(R_j\) is less than the Modification Rate (MR) are modified. Therefore, Eq. 4 must be replaced with Eq. 7 under this consideration:
\[
v_{ij} = \begin{cases} 
\phi_j(m_{ij} - m_{ij}), & R_{ij} < MR \\
m_{ij}, & \text{otherwise} 
\end{cases}
\] (7)
where \(\phi_j\) is a random number in the range of [-SF, SF] while it is in the range of [-1,1] in the standard ABC. This variety in the range is the second main difference of the Modified ABC. SF is a control parameter called scaling factor and it is automatically tuned according to Rechenberg’s 1/5 mutation rule during the search as in Eq.8:
\[
\begin{aligned}
SF_{i+1} = & \begin{cases} 
SF_i \ast 0.85, & \phi_m < 1/5 \\
SF_i / 0.85, & \phi_m > 1/5 \\
SF_i, & \phi_m = 1/5 
\end{cases} \\
\end{aligned}
\] (8)
In this study, we applied the modified ABC to inverse kinematics problem of 6-DOF PUMA 560 robot arm with a proposed cost function, and calibrated the parameters of algorithm by using variance analysis and Duncan’s multiply range test.

### 4. Proposed Approach

Here, the optimization problem is to find the optimum angle value \(\theta\) for each joint with the given initial Cartesian coordinate \((x_0, y_0, z_0)\) and the target coordinate \((x, y, z)\), so the end-effector of robot arm is transferred to desired location by \(\theta\). Obviously, the accurate calculations of \(\theta\) values are very important. The main goal of this study is to solve this optimization problem by implementing the modified ABC. For this purpose, we designed a cost function described in the next section.

### 4.1. Cost Function of 6-DOF Robot Arm

First, the current joint angles \(\theta'\) is transformed to the current Cartesian coordinate \((x', y', z')\) by using forward kinematics of PUMA 560 defined in Eq. 3. The cost function \(c\) is based on Euclidean distance between the desired location \((x, y, z)\) and the current location \((x', y', z')\). This cost can be used to calculate fitness function. The Pseudo code of the cost function is given below:

```java
// Initial Parameters of PUMA 560 Robot Arm
1 Set the parameters a2, a3, d3, d4 in table 1
2 Calculate the transformation matrix \(^6T_6\)
```
3 Produce new coordinate \((x', y', z')\) by multiplying of \(\theta^T_i\) and \((x_i, y_i, z_i)\)

4 Calculate distance of two location as 
\[ c = \sqrt{(x' - x)^2 + (y' - y)^2 + (z' - z)^2} \]

Fitness function is used to find the optimum value of \(\theta\) which has the minimum fitness value.

### 4.2. PSEUDO CODE OF THE PROPOSED APPROACH

First, the approach starts with generating \(SN\) set of \(\theta\) corresponding to \(SN\) food sources in the modified ABC as seen in Eq. 9. The following Pseudo code shows the phases of modified ABC to obtain the minimum \(\theta_i\):

\[
\text{FoodSource}_i = [\theta_i \mid \theta_i = \text{rand}(lb, ub), \forall j = 1...D] \quad (9)
\]

1 // Initial Phase
2 Set the parameters \(SN\), \(limit\), maximum cycle number \(MCN\), \(MR\) and \(SF\)
3 Initialize the population of solutions \(m_i\) for \(i = 1...SN\) as in Eq. 9 and set \(trial_i = 0\)
4 Calculate \(SN\) fitness values of population
5 Set \(cycle = 1\)
6 Repeat
7 // Employed phase
8 For \(i = 1\) to \(SN\) do
9 \quad Generate a random number as neighbor food source \(m_i\)
10 \quad Generate a random number as \(R_j\) for each parameter of the food source \(m_i\) produce a new food source \(v_i\) for the employed bee by using Eq. 8 and evaluate its quality
11 \quad Apply a greedy selection process between \(v_i\) and \(m_i\) and select the better one
12 \quad If solution \(m_i\) does not improve, \(trial_i = trial_i + 1\); otherwise \(trial_i = 0\)
13 \quad end for
14 \quad Calculate the probability values \(p_i\) by Eq. 6 for each solutions using fitness values
15 // Onlooker Phase
16 \quad \(t = 0\), \(i = 1\), \(rand_i = \text{random}\)
17 Repeat
18 \quad if \(rand_i < p_i\) then
19 \quad \quad Generate a random number as \(R_j\) for each parameter of the food source \(m_i\) produce a new food source \(v_i\) for the employed bee by using Eq. 8 and evaluate its quality
20 \quad \quad Apply a greedy selection process between \(v_i\) and \(m_i\) and select the better one
21 \quad \quad If solution \(m_i\) does not improve, \(trial_i = trial_i + 1\); otherwise \(trial_i = 0\)
22 \quad \quad \(t = t + 1\)
23 \quad \quad \(rand_i = \text{random}\)
24 \quad \quad \(i = i + 1\)
25 \quad End if
26 \quad until \((t = SN)\)
27 // Scout Phase
28 if \(\text{max}(trial) > \text{limit}\) then
29 \quad Replace \(m_i\) with a new randomly produced solution by Eq. 9
30 \quad End if
31 \quad Memorize the best solution achieved so far
32 \quad \(cycle = cycle + 1\)
33 \quad Update amount of \(SF\) according to Eq. 8
34 Until \((cycle = MCN)\)

The initial phase of the Pseudo code has been explained in the Section 3.1. In line 3, according to Eq. 5, \(SN\) fitness values are computed from \(SN\) cost functions which are calculated from \(SN\) set of \(\theta\). The employed phase of the Pseudo code has been discussed in the Section 3.2. Line 12 has been detailed in the Section 3.3. The onlooker phase of the Pseudo code has been given in the Section 3.4. The Section 3.5 explains the scout phase of the Pseudo code.

### 5. SIMULATION AND RESULTS

The proposed approach has been simulated in MATLAB environment. The solution time of the proposed approach is inversely proportional to the position error of end-effector. The parameters which are population number \((NP)\), maximum cycle number \((MCN)\), \(limit\) and modification rate \((MR)\) in the modified ABC affect the performance of the proposed approach. To obtain trade-off parameters, we applied multi-variance analysis and Duncan’s multiply range test by running simulation 5600 times with 560 sets of parameters. The results of Duncan’s tests are given in table 2.

| Table 2. The Result of Duncan's Multiply Range Test. |
|-----------------|-----------------|-----------------|-----------------|-----------------|
|                | \(NP\)           | \(MCN\)          | \(limit\)       | \(MR\)          |
| Value          | Mean            | Value           | Mean           | Value           | Mean           |
| Position error | 30 0.0000*      | 200 0.0000*     | 80 0.0001*     | 0.3 0.0001*     |
|                | 16 0.0000*      | 150 0.0000*     | 100 0.0001*    | 0.9 0.0001*     |
|                | 20 0.0001*      | 100 0.0001*     | 60 0.0001*     | 0.4 0.0001*     |
|                | 26 0.0001*      | 50 0.0002*      | 40 0.0001*     | 0.5 0.0001*     |
|                | 10 0.0002*      |                |                | 0.6 0.0001*     |
|                | 10 0.0002*      |                |                | 0.7 0.0001*     |
|                | 10 0.0002*      |                |                | 0.8 0.0001*     |

| Solution time  | 10 0.1000*      | 50 0.1028*      | 60 0.2409*     | 0.9 0.2365*     |
|                | 26 0.1956*      | 100 0.1990*     | 80 0.2499*     | 0.8 0.2388*     |
|                | 20 0.2431*      | 150 0.2967*     | 100 0.2462*    | 0.3 0.2418*     |
|                | 16 0.3117*      | 200 0.3823*     | 40 0.2477*     | 0.7 0.2468*     |
|                | 30 0.3776*      |                |                | 0.4 0.2478*     |
|                | 0.2509*         |                |                | 0.5 0.2509*     |
|                | 0.2538*         |                |                | 0.6 0.2538*     |

where a is the best subset and e is the worst subset.

The subset of \(NP=16\) is “a” (error) and “d” (time). The subset of \(MCN=150\) is “b” (error) and “c” (time). The subset of \(limit=60\) is “a” (both error and time). The subset of \(MR=0.9\) is “a” (both error and time). Here, the performance criterion was chosen as the minimum position error of the end-effector. So we assume the parameters of the approach as given in table 3.
The proposed approach was run 1000 times under the parameters mentioned in Table 3. Table 4 gives a comparison of the mean position errors and solution times achieved by the proposed approach, Particle Swarm Optimization, and Harmony Search. As seen in Table 4, the proposed approach offers both more accurate positioning and the less solution time than the other heuristic methods. To decrease solution time more, all the parameters of the approach in Table 2 may be chosen in “a” subset according to the solution time. The proposed method is adaptive, and it can be adjusted to many systems whose requirements may differ.

Table.3. Considered Value for Parameters.

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>16</td>
</tr>
<tr>
<td>SN</td>
<td>13</td>
</tr>
<tr>
<td>MR</td>
<td>0.9</td>
</tr>
<tr>
<td>MCN</td>
<td>150</td>
</tr>
<tr>
<td>limit</td>
<td>60</td>
</tr>
<tr>
<td>D (Dimension)</td>
<td>6</td>
</tr>
<tr>
<td>SF</td>
<td>2 (in beginning)</td>
</tr>
</tbody>
</table>

The proposed approach was run 1000 times under the parameters mentioned in Table 3. Table 4 gives a comparison of the mean position errors and solution times achieved by the proposed approach, Particle Swarm Optimization, and Harmony Search. As seen in Table 4, the proposed approach offers both more accurate positioning and the less solution time than the other heuristic methods. To decrease solution time more, all the parameters of the approach in Table 2 may be chosen in “a” subset according to the solution time. The proposed method is adaptive, and it can be adjusted to many systems whose requirements may differ.

Table.4. Performance Comparison of Heuristic Methods for Inverse Kinematics.

<table>
<thead>
<tr>
<th>Proposed Approach</th>
<th>PSO</th>
<th>HS</th>
</tr>
</thead>
<tbody>
<tr>
<td>position error</td>
<td>6.31059E-13</td>
<td>3.32E-08</td>
</tr>
<tr>
<td>solution time</td>
<td>0.028618442</td>
<td>0.0361</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

In this paper, a new heuristic approach based on the modified Artificial Bee Colony algorithm has been proposed for the inverse kinematics problem of robot arms, and it has been simulated on PUMA 560. It was compared with the other heuristic algorithm: Particle Swarm Optimization and Harmony Search. The simulations show the proposed approach gives better performance in terms of both the positioning error and the solution time than the other algorithms. Here, although the parameters of the approach have been chosen to obtain the minimum position error, the approach has produced the solution in less time than the other algorithms. The parameters can be changed to achieve the minimal solution time or the position error according to the performance requirements of the system. Therefore, the proposed approach is a candidate method to solve inverse kinematics problem of robot arms with the high numbers of joint especially.

REFERENCES