Solving Inverse Kinematics of Robot Manipulators by Means of Meta-Heuristic Optimisation

Kritsada Wichapong, Sujin Bureerat and Nantiwat Pholdee *

Sustainable and Infrastructure Research and Development Center, Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen, 40002, Thailand.

*Corresponding author: nantiwat@kku.ac.th

Abstract. This paper presents the use of meta-heuristic algorithms (MHs) for solving inverse kinematics of robot manipulators based on using forward kinematic. Design variables are joint angular displacements used to move a robot end-effector to the target in the Cartesian space while the design problem is posed to minimize error between target points and the positions of the robot end-effector. The problem is said to be a dynamic problem as the target points always changed by a robot user. Several well established MHs are used to solve the problem and the results obtained from using different meta-heuristics are compared based on the end-effector error and searching speed of the algorithms. From the study, the best performer will be obtained for setting as the baseline for future development of MH-based inverse kinematic solving.

1. Introduction
Robot manipulators are widely used in production lines for the purpose of non-stop lines, high accuracy lines, etc. To control a robot position, a mapping function from the Cartesian space to joint space (also known as inverse kinematic problem) is necessary. Solving such an inverse kinematic problem for a redundant robot is said to be the issue in the field of robotics as singularity can occur in robot position analysis when using conventional methods such as a geometric method [1], and an algebraic method [2-3]. In addition, a classical iterative method [4] is time consuming and the solution is not guaranteed as it depends on an initial solution. In this regards, many researchers have studied alternative ways to deal with the kinematic problem such as using artificial neural networks (ANN) [5-7], using the combination of artificial neural networks and meta-heuristics [8-10]. However, using ANN still has some disadvantages in long training time and less accuracy. As a result, seeking for an alternative strategy to solve the inverse kinematic problem of a redundant robot is interesting and challenging. Using MHs alone for tackling the inverse kinematic problem based on forward kinematics seems to be reasonable as the singularity problem can be avoided. It is flexible without requirements of training data.

The MH is a global search optimization method which has advantages in its derivative-free feature; therefore, they can deal with all kind of objective functions and design variables. It can explore a Pareto front within a single run for multiobjective optimization. As a results, the MHs is widely applied to various applications such as structural optimization [11], flow shop scheduling [12-13], a traveling sale man problem [14], network design [15], damage detection [16], etc. They are also
applied to solve the inverse kinematic problem of redundant robots based on solving forward kinematics by defining the optimization problem to find the joint variables attaining Cartesian points of the end-effector closest to the target points [9]. However, due to disadvantages of search convergence rate and consistency of MHs, using MHs for solving the inverse kinematic problem of redundant robots is rare. Development of MHs with acceptable convergence rate and consistency for solving the inverse kinematic problem is always useful. Also, comparative studies of various well established MHs for solving such an inverse problem should be investigated since it will initiate the baseline for MH performance in solving the problem.

In this work, a comparative study of several well established MHs for solving the inverse kinematic problem of a 4D redundant robot based on using forward kinematics as proposed in [9] is conducted. Design variables determine robot joint angular positions which can attain Cartesian points at the target points after solving the forward kinematics. The objective is posed to minimize a distance error between the target points and the calculated points obtained from the given joint angles. Several well established MHs including Differential evolution (DE) [17], Artificial Bee Colony (ABC) [18], Adaptive Differential evolution (JADE) [19], Teaching–learning-based optimization (TLBO) [20], Real-code ant colony optimization (ACOR) [21] and Population-based incremental learning (PBIL) [22 -23] are used to solve the problem for a verity of target points. The results obtained are compared in terms of search speed and accuracy.

2. Kinematic analysis of robotic manipulators

Robot manipulators are constructed by connecting joints and links which are modelled as an open-loop chain mechanism. To control them, kinematic analysis to understand the motion of the robot with respect to a fixed reference is necessary. Normally, kinematic analysis of a robot can be divided into forward and inverse kinematics. The forward kinematic is a problem in which joint angular positions are pre-specified. The end-effectors positions in Cartesian space are then found to meet the given values of the joint angles. The problem is expressed in Equation (1)

$$F_{\text{forward kinematic}}(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5) = (X, Y, Z, \phi_x, \phi_y, \phi_z)$$  \hspace{1cm} (1)

where $\theta_i$ are joint angles and X, Y, Z, are position of end-effectors in X, Y, and Z, direction. $\phi_x, \phi_y, \phi_z$ are end-effectors orientations in X, Y and Z, direction, respectively.

The inverse kinematic problem on the other hand is the reverse of the forward kinematic which can be defined as Equation (2).

$$F_{\text{inverse kinematic}}(X, Y, Z, \phi_x, \phi_y, \phi_z) = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$$  \hspace{1cm} (2)

To solve the forward kinematic problem, the Denavit-Hartenberg function (DH) is usually applied with ease while the inverse kinematic problem still has an issue as mentioned earlier.

In this study, MHs are applied to solve the inverse kinematic problem based on using the forward kinematics. A four-DOF robot model as presented in [9] is employed. The schematic diagram of the robot is shown in Figure 1. To solve the forward kinematic of the robot, DH is applied. The DH transformation matrix can be expressed as

$$i = (\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)Q_i(d_i) =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_i & -s_i & 0 & 0 \\ s_i & c_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$  \hspace{1cm} (3)
where \( a_i \) is distance from \( z_{i-1} \) to \( z_i \) along \( x_i \), \( \alpha_i \) is angle between \( z_{i-1} \) to \( z_i \) measure in plane normal to \( x_i \), \( d_i \) is the distance from the origin of the \((i-1)\) coordinate frame to the intersection of the \( x_{i-1} \) axis along the \( z_{i-1} \) axis, \( \theta_i \). is the joint from \( x_{i-1} \) to \( x_i \).

For the four-DOF robot, the DH variables are shown in Table 1 while transformation matrices of each link can be expressed as the following equations (5-8).

\[
0^T = \begin{bmatrix}
    c_1 & -s_1 & 0 & 0 & a_{i-1} \\
    s_1 & c_1 & 0 & 0 & \\
    0 & 0 & 1 & L_i & \\
    0 & 0 & 0 & 1 & 
\end{bmatrix}
\]

(4)

\[
1^T = \begin{bmatrix}
    c_2 & -s_2 & 0 & 0 & \\
    s_2 & c_2 & 0 & 0 & \\
    0 & 0 & 1 & 0 & \\
    0 & 0 & 0 & 1 & 
\end{bmatrix}
\]

(5)

\[
2^T = \begin{bmatrix}
    c_3 & -s_3 & 0 & L_2 & \\
    s_3 & c_3 & 0 & 0 & \\
    0 & 0 & 1 & 0 & \\
    0 & 0 & 0 & 1 & 
\end{bmatrix}
\]

(6)

\[
3^T = \begin{bmatrix}
    c_4 & -s_4 & 0 & L_3 & \\
    s_4 & c_4 & 0 & 0 & \\
    0 & 0 & 1 & 0 & \\
    0 & 0 & 0 & 1 & 
\end{bmatrix}
\]

(7)

\[
4^T = \begin{bmatrix}
    1 & 0 & 0 & L_4 & \\
    0 & 0 & -1 & -L_5 & \\
    0 & 1 & 0 & 0 & \\
    0 & 0 & 0 & 1 & 
\end{bmatrix}
\]

(8)

As a consequence, the transformation matrices of the end-effector can be determined as

\[
^e_T = \prod_{i=1}^{4}^iT^T = \begin{bmatrix}
    n_x & s_x & a_x & p_x \\
    n_y & s_y & a_y & p_y \\
    n_z & s_z & a_z & p_z \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]

(9)
Figure 1. The schematic diagram of the 4D robot

Table 1. D-H parameter of the 4-D robot [9]

| i  | $a_{i-1}$ (mm) | $\alpha_{i-1}$ (°) | $d_i$ (mm) | $\theta_{i-1}$ (°) |
|----|----------------|-----------------|-----------|------------------|
| 1  | 0              | 0               | 80        | $\theta_1$       |
| 2  | 0              | -90             | 0         | $\theta_2$       |
| 3  | 120            | 0               | 0         | $\theta_3$       |
| 4  | 65             | 0               | 0         | $\theta_4$       |
| 5  | 85             | 90              | 25        | 0                |

The positions and orientations of the end-effector are shown in Equation (10). $P_x$, $P_y$, and $P_z$ are position of end-effectors in X, Y, and Z, direction while $n_x$, $s_x$, $a_x$, $n_y$, $s_y$, $a_y$, $n_z$, $s_z$, $a_z$, are rotation component of $\phi_x$, $\phi_y$, $\phi_z$. To solve the inverse kinematic problem by using a meta-heuristic optimizer, the objective function can be formulated as

$$\text{Min: } \text{Error} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Subject to

$$-90 < \theta_1 < 90$$
$$-180 < \theta_2 < 0$$
$$0 < \theta_3 < 145$$
$$-90 < \theta_4 < 90$$

3. Meta-heuristic

In this work, six well established MHs are compared based on solving the inverse kinematic optimization problem detailed in the previous section. Given that $n_P$ is a population size (unless otherwise specified) and $n$ is a number of design variables, those MHs and their optimization parameter settings (details of notations can be found in the corresponding reference of each method) are detailed as:

- Differential evolution (DE) (1): The DE/best/2/bin strategy was used. A scaling factor, crossover rate and probability of choosing elements of mutant vectors are 0.5, 0.7, and 0.8 respectively.
- Artificial bee colony algorithm (ABC) (2): The number of food sources for employed bees is set to be \( n_p/2 \). A trial counter to discard a food source is 150.
- Adaptive differential evolution with optional external archive (JADE) (3): Differential evolution with composite trial vector generation strategies and control parameters.
- Teaching-learning-based optimization (TLBO) (4): Parameter settings are not required.
- Real-code ant colony optimization (ACOR) (5): The parameter settings are \( \theta = 0.2 \), and \( \xi = 1 \).
- Population-based incremental learning (BPBIL) (6): The learning rate, mutation shift, and mutation rate are set as 0.5, 0.7, and 0.2 respectively.

To verify the search performance in terms of speed and accuracy over the Cartesian space of the robot, 30 testing points for the inverse kinematic problem are assigned leading to be a dynamic problem. The testing points are generated randomly by means of a Latin Hyper Cube sampling technique (LHS). The population size is set to be \( n_p = 80 \) whereas the number of iterations is 250 for all test problems. The termination criteria are set to stop the search when the objective function value (Error) is lower than \( 1 \times 10^{-3} \) or the total number of function evaluations is equal to \( 80 \times 250 \) for all optimizers. The penalty function used in this study is a fuzzy set penalty function as detailed in [8].

**4. Result**

After performing optimization of the 30 points inverse kinematic problems, the results are shown in Tables 2-3. The accuracy of the MHs is measured based on position error of end-effectors as reported in Table 2 while the search speed of the MHs is measured based on number of function evaluations required to reach the target position as reported in Table 3. The lower position error is the more accurate optimizer while the lower number of function evaluations is the faster optimizer. In addition, the total number of successful runs from 30 points inverse kinematic problems is used to measure the search consistency, the more number of successful runs mean the corresponding method is more consistent. Note that, the algorithm that is terminated by the total number of function evaluations reaching the predetermined maximum evaluation number without giving acceptable position error is considered unusable.

From Table 2, the best optimizer based on average position error is TLBO while the second best is DE. When looking at the search consistency based on the number of successful runs, the most consistent optimizer is TLBO while the second best is DE similar to the measure of accuracy. For the search speed as reported in Table 3, the fastest optimizer on solving robot inverse kinematic problems is TLBO which require average 10405 function evaluations to meet the target position. The second best search speed is DE which average 11027 function evaluations to meet the target position. From the table, it was found that only TLBO and DE is usable for solving robot inverse kinematic problems since ABC, JADE, ACOR and BPBIL cannot meet target positions until maximum number of function evaluations (20,000) is reached.

| Point | DE       | ABC     | JADE    | TLBO    | ACOR    | BPBIL   |
|-------|----------|---------|---------|---------|---------|---------|
| no.1  | 8.1893x10^-6 | 2.0890x10^-1 | 3.0283x10^4 | 7.0879x10^6 | 1.5728x10^1 | 9.9300  |
| no.2  | 8.5071x10^-6 | 1.6049x10^-1 | 1.0078x10^3 | 9.4869x10^6 | 5.8601x10^4 | 9.3025x10^4 |
| no.3  | 6.2927x10^-1 | 1.3235x10^-1 | 1.3056x10^2 | 7.4932x10^6 | 6.3606x10^4 | 3.5103  |
| no.4  | 4.0340x10^-6 | 7.2377x10^-2 | 2.7022x10^5 | 6.7992x10^6 | 2.0240x10^-2 | 5.7221  |
| no.5  | 9.1439x10^-6 | 2.0680x10^-2 | 7.0457x10^8 | 8.3479x10^6 | 4.9800x10^2 | 9.8531  |
| no.6  | 1.7055    | 2.1507x10^-1 | 1.7058    | 1.7055    | 1.7055    | 2.5967  |
| no.7  | 7.4033x10^-6 | 3.9507x10^-2 | 3.4879x10^-2 | 2.2785x10^6 | 1.5208x10^-4 | 6.1441  |
| no.8  | 1.4250x10   | 3.9924x10^-1 | 7.0118x10^2 | 4.8653x10^6 | 8.3730x10^-4 | 4.1008  |
| no.9  | 2.7551x10^-6 | 7.2261x10^-1 | 2.4393x10^-2 | 9.1132x10^6 | 1.3526x10^-1 | 1.2252x10^-1 |

Table 2. Position error (Each cell is the average value from 30 runs)
Successful
Average
no.30
no.29
no.28
no.27
no.25
no.24
no.23
no.22
no.21
no.20
no.19
no.18
no.17
no.16
no.15
no.14
no.13
no.12
no.11
no.10
no.19
no.18
no.17
no.16
no.15
no.14
no.13
no.12
no.11
no.10

Table 3. Average number of function evaluations

| Point | DE      | ABC      | JADE     | TLBO     | ACOR     | BBPIL    |
|-------|---------|----------|----------|----------|----------|----------|
| no.1  | 9280    | 20000    | 20000    | 11040    | 20000    | 20000    |
| no.2  | 8960    | 20000    | 20000    | 9440     | 20000    | 20000    |
| no.3  | 20000   | 20000    | 20000    | 8960     | 20000    | 20000    |
| no.4  | 9360    | 20000    | 20000    | 12800    | 20000    | 20000    |
| no.5  | 8800    | 20000    | 20000    | 7520     | 20000    | 20000    |
| no.6  | 20000   | 20000    | 20000    | 20000    | 20000    | 20000    |
| no.7  | 7360    | 20000    | 20000    | 11200    | 20000    | 20000    |
| no.8  | 20000   | 20000    | 20000    | 10400    | 20000    | 20000    |
| no.9  | 9120    | 20000    | 20000    | 10240    | 20000    | 20000    |
| no.10 | 10240   | 20000    | 20000    | 9120     | 20000    | 20000    |
| no.11 | 9440    | 20000    | 20000    | 10080    | 20000    | 20000    |
| no.12 | 8880    | 20000    | 20000    | 10880    | 20000    | 20000    |
| no.13 | 9920    | 20000    | 20000    | 20000    | 20000    | 20000    |
| no.14 | 8320    | 20000    | 20000    | 9280     | 20000    | 20000    |
| no.15 | 9520    | 20000    | 20000    | 8000     | 20000    | 20000    |
| no.  | Value 1 | Value 2 | Value 3 | Value 4 | Value 5 | Value 6 |
|------|---------|---------|---------|---------|---------|---------|
| no.16| 9120    | 20000   | 20000   | 7360    | 20000   | 20000   |
| no.17| 7280    | 20000   | 20000   | 7840    | 20000   | 20000   |
| no.18| 10240   | 20000   | 20000   | 7360    | 20000   | 20000   |
| no.19| 10160   | 20000   | 20000   | 10560   | 20000   | 20000   |
| no.20| 20000   | 20000   | 20000   | 10400   | 20000   | 20000   |
| no.21| 20000   | 20000   | 20000   | 20000   | 20000   | 20000   |
| no.22| 8080    | 20000   | 20000   | 8320    | 20000   | 20000   |
| no.23| 10160   | 20000   | 20000   | 9440    | 20000   | 20000   |
| no.24| 8880    | 20000   | 20000   | 9920    | 20000   | 20000   |
| no.25| 8320    | 20000   | 20000   | 10720   | 20000   | 20000   |
| no.26| 10400   | 20000   | 20000   | 7520    | 20000   | 20000   |
| no.27| 8800    | 20000   | 20000   | 8960    | 20000   | 20000   |
| no.28| 9440    | 20000   | 20000   | 7200    | 20000   | 20000   |
| no.29| 9440    | 20000   | 20000   | 8800    | 20000   | 20000   |
| no.30| 11280   | 20000   | 20000   | 8800    | 20000   | 20000   |
| Average| 11027 | 20000 | 20000 | 10405 | 20000 | 20000 |

Figure 2 shows the search history for the best performer of all algorithms while Figure 3 shows the zoom-in of Figure 2. From these figures, it was found that, although all algorithms have fast convergence in the beginning, most of them are trapped at some points except for the DE and TLBO. The DE can converge to the optimum result by using lower than 1000 function evaluations whereas the TLBO can converge to the optimum result at about 3500 function evaluations.
5. Conclusions
In this work, a comparative study of meta-heuristic algorithms (MHs) for solving inverse kinematics of robot manipulators based on using forward kinematic is successfully investigated. The design variables are joint angular displacements used to move the robot end-effector to the target in the Cartesian space while the design problem is posed to minimize error between target points and the robot end-effector position. Six well established MHs are used to solve 30 testing points of inverse kinematic and the results are compared in terms of speed and accuracy. It was found that, the best performer in terms of both speed and accuracy is TLBO while the second best is DE. The others are said to lack on searching results within limitation of the total number of function evaluations as they cannot obtained the satisfy position error. It should be noted that this study is yet to be the real dynamic or on-line optimization problem as the target points are pre-defined before running an optimizer from scratch. With the use of additional numerical schemes such as an archiving technique, the performance of the two best optimizers could still be enhanced.

Acknowledgements
The research described in this paper was financially supported by the Thailand Research Funds (TRF).

References
[1] Ellwood R J, RaatZ J and Hesselbach A 2010 In: Precision Assembly Technologies and Systems, Chamonix, France 123-130
[2] Duffy J 1980 In: Analysis of Mechanisms and Robot Manipulators, Wiley, New York.
[3] Manocha D and Canny J F 1994 IEEE Transactions on Robotics and Automation. 10 648-57
[4] Korein J U and Badler N I 1982 IEEE Computer Graphics and Applications. 2 71-81
[5] Köker R 2004 Robotics and Autonomous Systems. 39 227-34
[6] Mao Z and Hsia T 1997 *Robotica* **15** 3-10
[7] Guo J and Cherkassky V 1989 *In: International 1989 Joint Conference on Neural Networks, Washington DC, USA* **2** 299-304
[8] Köker R 2013 *Information Sciences*. **222** 528-43
[9] Saka M P, Hasançebi O and Geem Z W 2016 *Swarm and Evolutionary Computation*. **28** 88-97
[10] Köker R 2016 *Engineering with Computers*. **32** 1-13
[11] Bureerat S and Pholdee N *Journal of Computing in Civil Engineering* **30** 04015019
[12] Liu B, Wang L and Jin Y H 2007 *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **37** 18-27
[13] Sangsawang C and Sethanan K 2016 *KKU ENGINEERING JOURNAL* **43** 55-61
[14] Nakkaew P, Kantanantha N and Wongthatsanekorn W 2016 *KKU ENGINEERING JOURNAL*, **43** 62-8
[15] Pepper J W and Golden B L 2002 *IEEE Transactions on Systems Man, and Cybernetics, Part A: Systems and Humans* **32** 72-77
[16] Pattnaik S B, Mohan S and Tom V M 1998 *Journal of Transportation Engineering* **124** 368-75
[17] Miguel L F F *Expert Systems with Applications* **39** 9704-14
[18] Storn R and Price K *Journal of Global Optimization* **11** 341-59
[19] Omran M G H, Salman A and Engelbrecht A P 2005 *In:Computational Intelligence and Security: International Conference, China* **15-19
[20] Rao R V, Savsani V J and Vakharia D P 2011 *Computer-Aided Design* **43** 303-15
[21] Socha K and Dorigo M 2008 *Journal of Operational Research* **185** 1155-73
[22] Baluja S 1994 *School of Computer Science, Carnegie Mellon University, Pittsburgh*
[23] Kunakote T and Bureerat S *KKU Research Journal* **19** 62-8