Trajectory Planning of a 6D Robot based on Meta Heuristic Algorithms

Kritsada Wichapong [1], Nantiwat Pholdee[1], Sujin Bureerat[1] and Thana Radpukdee [2]

[1]*Sustainable and Infrastructure Research and Development Center, Department of Mechancial Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen, Thailand*
[2]*Department of Industrial Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen, Thailand*

Abstract. In this work, several established meta-heuristics (MHs) were employed for solving 6-DOF robot trajectory planning. A fourth order polynomial function is used to represent a motion path of the robot from initial to final points while an optimisation problem is posed to minimise travelling time subject to velocity, acceleration and jerk constraints. The design variables are joint velocities and accelerations at intermediate positions, and moving time from the initial position to the intermediate position and from the intermediate position to the final position. Several MHs are used to solve the trajectory optimisation problem of robot manipulators while their performances are investigated. Based on this study, the best MH for robot trajectory planning is found while the results obtained from such a method are set as the baseline for further study of robot trajectory planning optimisation.

1 Introduction

Nowadays, robot manipulators are used to support or replace human in various applications such as industry, military, space, etc. To control motion of a robot, one of the most important things is robot trajectory planning used to generate referent input for the robot control system. The mission of trajectory planning is to find a geometric path of robot motion from initial to final positions under kinetic and dynamic constraints. Under working functions, the robot manipulator is expected to perform at the maximum performance such as minimising travelling time, minimising energy without damaging the system. Therefore, the robot trajectory planning is usually set to be an optimization problem to minimise travelling time [1], minimise energy [2] or minimise jerk [3] subject to position, velocity and acceleration constraints and one of the recently popular optimization techniques used to deal with this kind of problem is meta-heuristics (MHs) [4].

Over last decade, successful applications of MHs for robot manipulator trajectory planning have been reported worldwide. Due to the advantages of global search and without the use function derivatives, the MHs are popularly used for various real engineering optimization problems such as structural optimisation [5], flow shop scheduling [6,7], a traveling sale man problem [8], network topology design [9], damage detection [10], etc. Although, MHs are popular and widely used in various applications, the MHs have some disadvantages of search convergence rate and consistency. Development of MH for a particular problem are always required to get the best optimum results. For the robot trajectory plaining, applications of MHs were reported in [1-4]. Although, some MHs have been implemented with the trajectory, there are numerous MHs developed in the part few years and most of them have never been implemented. Therefore, investigation of those MHs performance for such a problem is worthwhile.

In this work, several established MHs are employed for solving 6-DOF robot manipulator trajectory planning. The objective function is assigned to minimise trajectory time subject to the constraints for velocity, acceleration, and jerk of all joints. The design variables are joint velocities and acceleration at intermediate positions, and moving time from initial to intermediate positions and from intermediate to final positions. The fourth order polynomial function is used to represent the motion equation [11]. Several MHs including, Differential evolution (DE) [12], Artificial Bee Colony (ABC) [13], Adaptive Differential evolution (JADE) [14], Teaching–learning-based optimization (TLBO) [15], Real-code ant colony optimization (ACOR) [16], Population-based incremental learning (PBIL) [17], Sine Cosine algorithm (SCA) [18], Moth-flame optimization algorithm (MFO) [19], Whale optimization algorithm (WAO) [20], Success-History Based Adaptive Differential Evolution (SHADE) [21], Adaptive Differential Evolution with Linear Population Size Reduction (LSHADE) [22] and Evolution Strategy with Covariance Matrix Adaptation (CMAES) [23] are used to solve the problem and their performance are investigated.

2 Formulation of trajectory plaining optimisation problem

Trajectory planning is a problem set to find a motion path including position, velocity and acceleration as a function of time from initial to final defined points. In this work, to find the motion path of the 6-DOF robot (Fig.1), the complete trajectory is divided into two paths (Fig.2); from initial to intermediate positions and from intermediate to final positions or the target point while the fourth order polynomial function is used as a motion equation for both path [11]. The initial joint angle and final joint angle are predefined coordinate positions (θ_i, θ_f) while velocity and acceleration of initial and final positions are set to be zero. The path from the initial to the intermediate positions will have the same velocity and acceleration as the path from the intermediate to the final positions for the continuity of motion.

For the first part, the fourth order trajectory to be used from the initial to the intermediate position is given by Equation (1)

$$\theta_{i,i+1}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 \qquad (1)$$

The a_0 ,…, a_4 are polynomial constant coefficients which can be determined as given in Equation (2)

$$\begin{gathered} \theta_i = a_0 , \\ \dot{\theta}_i = a_1 , \ \dot{\theta}_{i+1} = a_1 + 2a_2 t_1 + 3a_3 t_1^{\,2} + 4a_4 t_1^{\,3} \\ \ddot{\theta}_i = 2a_2 , \ \ddot{\theta}_{i+1} = 2a_2 + 6a_3 t_1 + 12a_4 t_1^{\,2} \end{gathered} \qquad (2)$$

where t_1 is the time from initial position (i) to intermediate position ($i+1$).

For the second part of the trajectory from intermediate to final points, a fourth order trajectory is used. The fourth order trajectory to be used from the intermediate to the final positions is given by Equation (3)

$$\theta_{i+1,f}(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 \qquad (3)$$

The b_0 ,…, b_4 are polynomial constant coefficients which can be determined as given in equation (4)

$$\begin{gathered} \theta_f = b_0 + b_1 t_2 + b_2 t_2^{\,2} + b_3 t_2^{\,3} + b_4 t_2^{\,4} \\ \dot{\theta}_{i+1} = b_1 , \ \dot{\theta}_f = b_1 + 2b_2 t_2 + 3b_3 t_2^{\,2} + 4b_4 t_2^{\,3} \\ \ddot{\theta}_{i+1} = 2b_2 , \ \ddot{\theta}_f = 2b_2 + 6b_3 t_2 + 12b_4 t_2^{\,2} \end{gathered} \qquad (4)$$

To determine the trajectory of robot manipulators, the optimisation problem is posed to find traveling time from initial to intermediate position, traveling time from intermediate to final position, intermediate velocity and acceleration which can expressed as:

$$\mathbf{x}=\{t_1, t_2, \theta_{i,j=1},\ldots, \theta_{i,j=n}, \theta_{f,j=1},\ldots, \theta_{f,j=n}, \dot{\theta}_{i+1,j=1},\ldots,\dot{\theta}_{i+1,j=n},\ddot{\theta}_{i+1,j=1},\ldots,\ddot{\theta}_{i+1,j=n}\}$$

where n is the number of robot joints which are set to be 6.

The objective function is to minimise travelling time subjected to velocity, acceleration and jerk constraints. The problem can be expressed as;

Min : Travelling times = t_1+t_2

Subject to

$$\dot{\theta}_j \le \omega_c$$
$$\ddot{\theta}_j \le \alpha_c$$
$$\dddot{\theta}_j \le J_c$$

where ω_c = angular velocity constraint for joint j

α_c = acceleration constraint for joint j

J_c = angular jerk constraint for joint j

Table 1 and Table 2 shown the predefined initial points and final points to be an example for finding trajectory and constraint values for each joints [1].

.

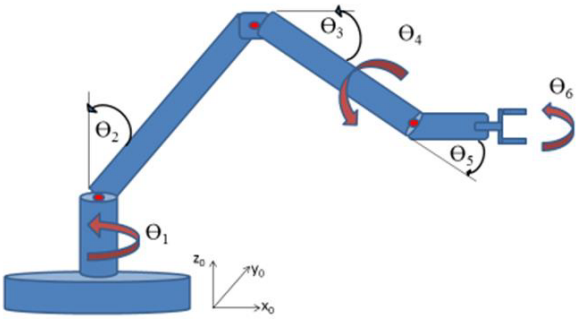

Figure 1. The schematic diagram of the 6D robot.

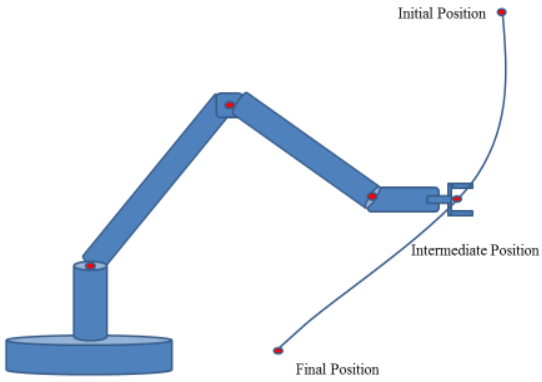

Figure 2. The 3 positions on the trajectory.

Table 1. Joint angular positions in each joint of the robot manipulator

Joint	θ_i (deg)	θ_f (deg)
1	-10	55
2	20	35
3	15	30
4	150	10
5	30	70
6	120	25

Table 2. Constraints for each joint

Joint	ω_c (deg/s)	α_c (deg/s^2)	J_c (deg/s^3)
1	100	60	60
2	95	60	66
3	100	75	85
4	150	70	70
5	130	90	75
6	110	80	70

3 Numerical experiment

In this work, twelve established MHs are used to solve the 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 [24] :

- Differential evolution (DE) [12]: 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) [13]: 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) [14]: Differential evolution with composite trial vector generation strategies and control parameters
- Teaching-learning-based optimization (TLBO) [15]: Parameter settings are not required.
- Real-code ant colony optimization (ACOR) [16]: The parameter settings are $q = 0.2$, and $\xi = 1$.
- Population-based incremental learning (BPBIL) [17]: The learning rate, mutation shift, and mutation rate are set as 0.5, 0.7, and 0.2 respectively.
- -Sine Cosine algorithm (SCA) [18]: The constant a parameter is set to be 2.
- Moth-flame optimization algorithm (MFO) [19]: The constant b parameter is set to be 1 while other parameters are iteratively adapted.
- Whale optimization algorithm (WOA) [20]: The b parameter is set to be 1 while other parameters are iteratively adapted.
- Success-History Based Adaptive Differential Evolution (SHADE) [21]: The parameters are self-adapted during an optimisation process.
- SHADE with Linear Population Size Reduction (L-SHADE) [22]: The parameters are self-adapted during an optimisation process.
- Evolution strategy with covariance matrix adaptation (CMAES) [23]: The parameters are self-adapted during an optimisation process.

Each optimiser is used to solve the problem for 20 optimisation runs. The population size is set to be n_P = 100 whereas the number of iterations is 100, which means the total number of function evaluations is equal to 80×250 for all optimizers. The penalty function used in this study is a fuzzy set penalty function as detailed in [25].

Table 3. Objective function values obtained from all optimisers

Optimiser	Mean	STD	Min	Max.
DE	3.2654	0.4517	2.5422	4.2544
ABC	5.4737	0.9690	3.7048	6.7106
TLBO	5.1052	0.5335	3.6935	5.8671
ACOR	5.5473	0.2954	5.0403	6.1818
BPBIL	7.8066	1.1301	6.1292	10.3227
SCA	8.1180	0.9280	6.7243	9.9459
MFO	5.4793	0.9677	3.5938	7.0451
WOA	7.6379	0.9168	6.0486	9.0543
JADE	1.4906	0.1609	1.1719	1.9356
SHADE	1.3362	0.3243	0.8401	1.8250
LSHADE	0.8108	0.3149	0.2099	1.4440
CMAES	0.4397	0.3589	0.2154	1.5547

Table 4. Design variables obtained from best run of the top five best optimisers

Design variables	DE	JADE	SHADE	LSHADE	CMAES
t_1	1.6011	0.5494	0.4478	0.0893	0.0429
t_2	0.9327	0.6225	0.3923	0.1206	0.1720
$\dot{\theta}_{i+1,1}$	0.0646	0.0130	-0.0223	0.0000	-0.0002
$\dot{\theta}_{i+1,2}$	-0.1666	0.0316	0.0139	-0.0016	-0.0009
$\dot{\theta}_{i+1,3}$	-0.0606	-0.0337	0.0138	0.0003	0.0012
$\dot{\theta}_{i+1,4}$	0.1019	-0.0097	-0.0224	0.0005	-0.0008
$\dot{\theta}_{i+1,5}$	0.0481	0.0129	0.0335	-0.0003	0.0001
$\dot{\theta}_{i+1,6}$	0.0115	0.0563	0.0310	0.0007	-0.0009
$\ddot{\theta}_{i+1,1}$	-0.0329	0.0429	0.0222	-0.0160	-0.0140
$\ddot{\theta}_{i+1,2}$	0.1110	0.0172	-0.0359	-0.0012	-0.0408
$\ddot{\theta}_{i+1,3}$	-0.1074	-0.1175	-0.0519	-0.0264	0.0537
$\ddot{\theta}_{i+1,4}$	-0.3059	-0.0694	-0.0259	-0.0157	-0.0377
$\ddot{\theta}_{i+1,5}$	-0.3139	0.0878	-0.0288	-0.0341	-0.0065
$\ddot{\theta}_{i+1,6}$	0.2677	0.0545	-0.0019	0.0143	-0.0409
Minimum objective function	2.5338	1.1719	0.8401	0.2099	0.2149
Maximum constraints violation	0.0058	0.0001	-0.0247	-0.0007	0.0049

4 Results and discussion

After performing 10 optimization runs of 12 MHs on solving the 6-DOF robot trajectory optimisation problem, the results are shown in Tables 3-4. The convergence of

the MHs is measured based on mean values of the objective function (Mean) while the consistency of the MHs is measured based on standard deviation values of the objective function (STD) as reported in Table 3. The lower Mean is the better convergence while the lower STD is the better consistency. From Table 3, the best convergence rate is obtained from using CMAES while the second best and the third best are LSHADE and SHADE, respectively. For the measure of search consistency, the most consistent is JADE while the second best and the third best in terms of consistency are ACOR and LSHADE, respectively. LSHADE obtained the lowest minimum objective function values while CMAES obtained the lowest maximum objective function value. Table 4 shown design variables, objective function and constraints violation of the best run for the top five best optimisers.

Based on this study, with a limit of total number of function evaluations, the algorithm with self-adaptive optimisation parameters perform better than the algorithms that require predefined optimisation parameters. The CMAES and LSHADE are said to be the top best MH for solving 6DOF robot trajectory plaining optimisation.

5 Conclusions

In this work, a comparative study of 12 MHs on solving 6-DOF robot trajectory plaining optimisation is conducted. The fourth order polynomial function is used to represent a motion path of the robot from initial to final points while the optimisation problem is posed to minimize travelling time subject to velocity, acceleration and jerk constraints. The results are compared in term of search convergence and search consistency. It was found that, CMAES gives the best convergence rate while JADE is the most consistent. LSHADE obtains the lowest minimum objective function. Among the 12 MHs, the CMAES and LHSHADE are said to be the top 2 best performers for such a problem. For the future work, development of MH in which optimisaiton parameter is self-adaption for the 6-DOF robot trajectory plaining optimisation will be carried out.

Acknowledgment

The authors are graceful for the support from the Thailand Research Fund (TRF)

References

1. P. Tangpattanakul, P. Artrit, ECTI-CON, (2009)
2. D. P. Garg, M. Kumar, Appl. Artif. Int **15**, 241-252 (2002)
3. A. Piazzi, A. Visioli, IEEE Trans. Ind. Elec **47**, 140-49 (2000).
4. T. T. Mac, C. Copot, D. T. Tran, R. De Keyser, Robot Auton. Syst **86**, 13-28 (2016).
5. S. Bureerat, and N. Pholdee, J. Comput. Civil Eng **30**, 04015019 (2016).
6. B. Liu, L. Wang, Y. H. Jin, IEEE T. Syst. Man Cyb.- Part B (Cybernetics), **37**, 18-27 (2007).
7. C. Sangsawang, K. Sethanan, KKU Eng. J **43**, 55-61 (2016).
8. J. W. Pepper, B. L. Golden, , IEEE T. Syst. Man Cyb.- Part A, **32**, 72-77, (2002).
9. S. B. Pattnaik, S. Mohan, V. M. Tom, J. Transp. Eng **124**, 368-375 (1998).
10. L.F.F.Miguel, Expert Syst. Appl **39**, 9704-9714, (2012).
11. P. Savsani, R. L. Jhala, V. J. Savsani, In Proc IEEE International Systems Conference (SysCon), pp. 381-386 (2013)
12. R. Storn, K. Price, J. Global Optim **11**, 341-359 (1997).
13. D. Karaboga, B.Basturk, J. Global Optim **39**, 459-471 (2007).
14. M.G.H. Omran, A. Salman, A.P. Engelbrecht, Computational Intelligence and Security: International Conference, China, , pp. 15-19 , (2005).
15. R.V. Rao, V.J. Savsani, D.P. Vakharia, Comput Aided Design **43**, 303-315 (2011).
16. K. Socha, M. Dorigo, Eur. J. Oper Res **185**, 1155-1173 (2008).
17. S. Baluja, School of Computer Science, Carnagie Mellon University: Pittsburgh, CMU_CS_95_163 (1994).
18. S. Mirjalili**,** Knowl.-Based Syst **96**, 120-133 (2016).
19. S. Mirjalili, Knowl.-Based Syst **89**, 228-49 (2015).
20. S. Mirjalili, A. Lewis, Adv. Eng. Softw **95**, 51-67 (2016).
21. R. Tanabe, A. S. Fukunaga, IEEE C. Evol. Computat (CEC2013), pp. 1952-1959 (2013).
22. R. Tanabe, A. S. Fukunaga, IEEE C. Evol. Computat (CEC2014), pp. 1658-1665, (2014).
23. N. Hansen, S. D. Muller, Evolutionary Compututation, **11**, 1-18 (2003).
24. K. Wichapong, S. Bureerat, N. Pholdee, IOP Conference Series: Materials Science and Engineering, 2018, in press.
25. N. Pholdee, S. Bureerat, Adv. Eng. Softw **75**, 1-13 (2014)