Trajectory Planning and Obstacle Avoidance Control of Redundant Robots Using Differential Evolution and Particle Swarm Optimization Algorithms

Sujan Warnakulasooriya[1] and S.G. Ponnambalam[2(✉)]

[1] MAS Unichela Pvt. Ltd., no. 124, Horana Road, Panadura, Srilanka
sujanmw@gmail.com

[2] Advanced Engineering Platform and School of Engineering, Monash University Malaysia, 46150 Bandar Sunway, Malaysia
sgponnambalam@monash.edu

Abstract. The problem of trajectory planning and obstacle avoidance in redundant robots is addressed in this paper. Four variants of Particle Swarm Optimization (PSO) and a Differential Evolution (DE) algorithm are proposed to solve this problem. Simulation experiments on a 5 degree-of-freedom (DOF) robot manipulator in an environment with static obstacles are conducted. The manipulator is required to move from a start position to a goal position with minimum error while avoiding collision with the obstacles in the workspace. The performance of the proposed algorithms is compared with the results reported in the literature and the comparative results are presented. It is observed that qPSO-C performs better in free space and PSO-C performs better in environment with obstacles in terms of minimizing error average convergence time. The performance of DE improves when the number of obstacles increases.

Keywords: Trajectory planning · Obstacle avoidance · Redundant robot · Differential evolution · Particle swarm optimization

1 Introduction

Robot manipulators are extensively used in repetitive tasks which require excellent accuracy and precision. These robot manipulators may have several degrees of freedom while it may also require avoiding collisions with obstacles which may exist in the surrounding environment. There are two common approaches for controlling such robot manipulators: forward kinematics and inverse kinematics. Forward kinematics approach can be better realized in joint space where the changes in joint angle values would immediately affect end-effector position. In contrary, inverse kinematics can be viewed as Cartesian space mapping problem where exact joint angle values are required to be computed according to the input of desired end-effector position. Thus heuristic approach may offer rich dividends in tackling this issue. Kim and Lee [1] proposed a hybrid algorithm with fuzzy logic and the procedure does not require solving the inverse kinematics of manipulators. Behesti and Tehrani [2] also used fuzzy

B.K. Panigrahi et al. (Eds.): SEMCCO 2014, LNCS 8947, pp. 596–605, 2015.
DOI: 10.1007/978-3-319-20294-5_51

logic concept and obstacle avoidance. Nearchou [3] proposed a genetic algorithm to determine the solution set of joint angles. Secară and Vlădăreanu [4] presented a strategy for obstacles avoidance of a redundant manipulator based on an iterative genetic algorithm. The objective of the strategy is to simultaneously minimize the end-effector location error and the manipulator total joint displacement while the collision with the obstacles is avoided. Zhang and Wang [5] developed a recurrent neural network and applied for kinematic control of redundant manipulators with obstacle avoidance capability. Differential Evolution [6] has also been used in robot manipulator problems [7, 8]. Particle Swarm Optimization which uses a swarm, moving in a continuously adjusting velocity towards the global best caught the eye of many researchers [9–11] as a possible route in achieving suitable algorithm for trajectory planning and obstacle avoidance. When enhancing the performance of the algorithm, issues such as avoiding singularities [12] and including a feedback using image processing was also addressed in [13, 14]. Goh and Ponnambalam [15] tested the performance of four variants of PSO for obstacle avoidance control of redundant robots.

The aim of this paper is to further improve the performance of the variants proposed by Goh and Ponnambalam [15] and also to propose a differential evolution algorithm. The paper is organized as follows. Section 2 presents the problem statement. Sections 3 and 4 present the implementation details of PSO and DE respectively. Section 5 details the parameter fine tuning for PSO and DE. Results and discussion are presented in Sect. 6. Conclusion is presented in Sect. 7.

2 Problem Statement

The objective of this paper is to propose DE and PSO to find Collision-free configuration of a 5-DOF planar redundant robot manipulator that allows the robot end effecter to travel from an initial position to the desired goal point with minimal error (greatest accuracy). The obstacles are static and presented as polygon in which closed loop of straight lines is formed. The performance of the proposed DE and the variants of PSO are evaluated.

2.1 Kinematics of a Robot Manipulator

A standard method introduced by Denavit and Hartenberg (D-H) [16] is useful to define joint matrices and link matrices to standardize the coordinate frames for spatial linkages. In the D-H convention, coordinate frames are attached to the joints between two links such that one transformation is associated with the joint and the second is associated with the link. This concept will allow the user to define the entire robot kinematics with a strict but yet simple representation. With respect to the robot manipulator, the corresponding D-H table is presented in Table 1. When the D-H table is available, devising the corresponding transformation matrix is a simple task. Take one row at a time and substitute appropriately those values to the standard transformation matrix which is given in Fig. 1.

Table 1. Denavit-Hartenberg table

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	l_1	0	θ_2
3	0	l_2	0	θ_3
4	0	l_3	0	θ_4
5	0	l_4	0	θ_5

$$ {}^{i-1}_{\ \ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Fig. 1. The standard transformation matrix

$$ \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & l_1 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & l_2 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & l_3 \\ s\theta_4 & c\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & l_4 \\ s\theta_5 & c\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} l_5 \\ 0 \\ 0 \\ 1 \end{bmatrix} $$

Fig. 2. Transformation matrices to obtain end effecter position

Using the above transformation matrix if all the link lengths and joint angles are known, by doing the matrix multiplication as shown in Fig. 2, position of the end effecter could be calculated.

3 Objective Function and Constraint

The main objective of this paper is to minimize the positional error between the end-effector and goal point. Also, the secondary objective is to satisfy the constraint of collision-free positioning of the robot manipulator. Considering the ability for robot to avoid collision with obstacle in workspace during movement to goal point, an additional weight called '*collision*' is added to the error equation. Considering the ability for robot to avoid collision with obstacle in workspace during movement to goal point, this is included in the objective function with the property shown in Eq. (1).

$$ collision = \begin{cases} 1, & \text{if collision - free movement} \\ 0, & \text{otherwise} \end{cases} \tag{1} $$

Simple logic is used to check if any of the links are colliding with any boundary of an obstacle. The process is carried out by iteratively selecting each link and by testing if the coordinates of the link cut through the area belongs to the obstacle. If it cuts through, then an additional weight is added to the error function which will result the specific joint angle combination to be disregarded during the search process.

The objective function is to minimize the error and to avoid collision. The objective function is shown in Eq. (2).

$$error = \sqrt{(Target_X - Current_X)^2 + (Target_Y - Current_Y)^2} + Collution \tag{2}$$

Where $Target_x$ and $Cuurent_x$ are the x-cordinastes of the goal position and the current position, and $Target_y$ and $Cuurent_y$ are the y-cordinastes of the goal position and the current position of the manipulator.

Singularities of the kinematic mapping, which determines the position of the end–effecter in terms of the manipulator's joint variables, may impede control algorithms, lead to large joint velocities, forces and torques and reduce instantaneous mobility. However they can also enable fine control, and the singularities exhibited by trajectories of the points in the end–effecter can be used to mechanical advantage.

Strictly speaking singularity can be simply defined as two links working as one link, which results the manipulator to have a lesser degree of freedom. Avoiding singularities is a necessity in a robotic manipulator.

Joint angle values are checked after each iteration and if the joint angle lie between −2.5 and 2.5°, it is randomly set to a different value to avoid singularities.

4 Differential Evolution

Differential Evolution [6] is a metaheuristic search algorithm tries to optimize the candidate solution space by iteratively making modifications to the existing solution. Even though the most optimum answer is not guaranteed, it is possible to obtain a solution with an acceptable error margin.

4.1 Generation of Initial Solution

Initial solution space generated is common to both PSO and DE algorithms. User can define the number of solution in a sample space but number of members in one solution is fixed as the numbers of joint angles which is fixed as five. Keeping practical aspects such as maximum and minimum servo angle, solution space should be randomly generated to accommodate the range of the servo motor.

Given below are sample strings (target vectors) with angles generated randomly within a specified range.

$$\begin{matrix}
[65.74 & 12.77 & -51.69 & 24.44 & 76.73] \\
[15.13 & -76.71 & -64.62 & 35.47 & 64.00] \\
. & . & . & . & . \\
[-60.85 & 41.96 & 71.60 & -45.14 & -14.49]
\end{matrix}$$

The three main steps in DE are Mutation, Crossover and Selection and the implementation details are explained below.

4.1.1 Mutation

Mutation operation is applied to every individual in the population. Equation (1) is used to do mutation.

$$M_a = I_{r1} + MF * (I_{r2} - I_{r3}) \tag{1}$$

Where, M_a is the donor vector, MF is the mutation factor and I_{r1}, I_{r12} and I_{r3} are randomly selected target vectors.

4.1.2 Crossover

Crossover operation is performed taking one joint angle at a time in to consideration. A random number is generated and if it is larger than the crossover factor, corresponding crossover vector point will be replaced by the donor vector point. Otherwise crossover vector point will use the value of the target vector point. Equations (2), (3) and (4) are used for crossover.

$$K = rand \rightarrow \in [0, 1] \tag{2}$$

$$K \geq CF, C_{a,b} = M_{a,b} \tag{3}$$

$$K < CF, C_{a,b} = I_{a,b} \tag{4}$$

Where, K is the random number, $M_{a,b}$ is the donor vector, $I_{a,b}$ is the target vector, CF is the crossover factor and $C_{a,b}$ is the trail Vector Point.

4.1.3 Selection

The selection operator of DE adopts a one-to-one competition between the target vector and the trial vector. If the objective function value of the trial vector is less than or equal to that of the target vector, then the trial vector will survive into the next generation, otherwise, the target vector will enter the next generation.

5 Particle Swarm Optimization

Particle Swarm Optimization [17] is an optimization method which uses an existing candidate solution iteratively to achieve the solution with the required quality. In PSO existing solution will move in the search space according to a simple mathematical formula over the particles position and velocity, up until it finds the optimal solution. In addition each particle will remember the best solution achieved (personnel best) and exchange information with other particles to determine the best solution (global best). Particles will move into a new position by adjusting its velocity in every generation. Thus the new position will be the sum of previous position and the current velocity.

In a standard PSO there are only two equations governing the performance. The first equation will update the velocity while second equation will update the position.

$$V_{id=}wV_{id} + C_1r_1(P_{ib} - X_{id}) + C_2r_2\left(P_{gb} - X_{id}\right) \tag{5}$$

$$X_{id} = X_{id} + V_{id} \tag{6}$$

V_{id} is the current/new velocity while X_{id} is the current/new position. w is the current velocity factor and r_1 and r_2 are random numbers distributed uniformly in [0, 1]. P_{ib} is the personnel best solution achieved by the selected particle whereas Pgb is the global best solution achieved by the whole Swarm. C_1 and C_2 are weights for personnel best and global best.

5.1 Variants of PSO

The four variants used by Chyan and Ponnambalam [14] are used in this paper. By conducting various experiments the performance of these four variants are improved. The performance improvement is by adopting suitable weights and by employing mechanisms to escape from local optima, if it happens. Reader can refer [14] for details of the four variants namely, namely PSO-C, PSO-W, qPSO-C and qPSO-W.

6 Parameter Fine Tuning for DE and PSO

Fine tuning parameters are an essential aspect as it aids immensely to enhance the performance of the respective algorithm. However deciding on what is the best parameter takes ample lot of time as many experiments have to be conducted. Nevertheless when an algorithm is fine-tuned perfectly, boost in performance can be quite rewarding. After conducting sensitivity analysis on the parameters, it is found that the mutation factor of 0.7 and the crossover factor of 0.5 perform better for DE algorithm implemented in this paper.

A rigorous analysis on the parameters for PSO-c and PSO-W variants is conducted and the parameters used in the paper are provided in table below. It is found that with these parameters and a local search approach implemented in the PSO variants, which is explained in Sect. 6.1 could obtain better results than the results reported in [14]. For the better understanding the parameter used in [14] and the optimal parameters found in this paper after the analysis conducted are presented in Table 2.

Table 2. Details of parameter fine tuning

Variant	Parameters	Parameters used in [15]		Parameter's used in this paper	
		Range tested	Optimal parameter	Range tested	Optimal parameter linearly decreasing from 0.9 to 0.4
All PSO variants	w	0.9 to 0.4	Linearly decreasing from 0.9 to 0.4 for first 100 iterations	0.9 to 0.4	Linearly decreasing from 0.9 to 0.4 for first 100 iterations
PSO-W	C_1	0.1 to 3	0.7	0.4 to 1.7	0.4
	C_2	0.1 to 3	1.0	0.4 to 1.7	0.6
PSO-C	C_1	2 to 6	5.8	2 to 4	2.4
	C_2	2 to 6	2.8	2 to 4	3.2
DE	Mutation factor	n/a	n/a	0.1 to 1.0	0.7
	Cross over factor	n/a	n/a	0.1 to 1.0	0.5

6.1 Strategies to Escape from Local Optima

One of the main issues of the metaheuristic search algorithms is that they converge too fast at times and due to this it may get stuck in local optima. Getting stuck in local optima may significantly reduce the performances of these algorithms. Identification of entrapment in local optima could be identified by a counter, where the counter is incremented by one when the quality of the solution is not improved. When this counter reaches a certain fixed value it could be concluded that the search process stuck in a local optima. If the solution gets stuck in local optima, it is difficult to move out from the local optima. To avoid this situation, an Iterated Local Search approach is adopted in this paper. The approach is to regenerate the solution space after a user defined number of counts when the process stuck in local optima. Another approach followed is the elite preserve strategy. Five best solutions in the previous generation is also randomly injected into the next generation. The PSO variants perform better with the optimal parameters and the strategies adopted in this paper. The other parameter values required for the variants are the same as in [15].

7 Experimental Conditions

The experimental conditions of the simulations conducted are detailed in this section.

The initial position of the manipulator is at X = Y = 0. The target position is X = Y = 22. The termination condition for all the algorithms is 500. Maximum error margin is set to 0.1. The population size is 60. The maximum loop count to start regeneration is set 50. Experiments are conducted with no obstacle, one obstacle and two obstacles in the environment. The obstacles are in the shape of rectangle with different positions.

The X-Y coordinates of the vertices of the obstacles are given below.

Obstacle-1 = [(−10, 8), (−10, 16), (11, 16), (11, 8), (−10, 8]

Obstacle-2 = [(28, 15), (28, 35), (38, 35), (38 15), (28, 15)]

8 Results and Discussion

The results of the experiments conducted in three scenarios are presented in this section. Each experiment is conducted ten time and the average values are reported. The measures used to evaluate the algorithms are average error, average converging time (in Sec) and average loop count. Table 3 shows the performance results of the algorithms in the environment with no obstacles.

In the above table performances of each of the algorithms is analyzed. A main criterion of concern is if algorithm can produce a result with an acceptable error margin. As it is illustrated from the above table all the algorithms will converge within the given margin. So in order to compare the algorithms, secondary criterions such as Converging Time and Loop Count should be taken in to consideration. Considering the loop count it is revealed from the above results that qPSO-C is the best algorithm closely followed by PSO-C for free space.

Table 4 shows the performance results of the algorithms in the environment with one obstacle.

It is evident from the results in Table 4 that all the algorithms converge within the acceptable error margin and thus secondary evaluation criterions should be considered. Above results for one obstacle space reveals a significant change with the results from the free space condition with PSO-C performs better over other algorithms.

Table 5 shows the performance results of the algorithms in the environment with two obstacles. Still PSO-C performs better over other algorithms. It is also observed that the performance of DE is improving as the number of obstacles increase.

Table 3. Performance of the algorithms in free space (no obstacles)

Free space	DE	PSO_C	PSO_W	qPSO_C	qPSO_W
Average error	0.0788	0.0589	0.0657	0.0790	0.0694
Average converging time (s)	0.4787	0.1724	0.3908	0.1832	0.3969
Average loop count	46.8	9.3	41.0	8.1	31.7

Table 4. Performance of the algorithms with one obstacle

One obstacle	DE	PSO_C	PSO_W	qPSO_C	qPSO_W
Average error	0.0715	0.0675	0.0742	0.0676	0.0632
Average converging time (s)	0.6028	0.1901	0.5501	0.2534	0.7313
Average loop count	42.0	7.1	35.4	8.7	48.8

Table 5. Performance results of the algorithms with two obstacles

Two obstacle	DE	PSO_C	PSO_W	qPSO_C	qPSO_W
Average error	0.0781	0.0669	0.0679	0.5080	0.0738
Average converging Time (s)	0.7093	0.2474	0.6803	0.2845	0.9858
Average loop count	37.4	7.7	35.6	8.3	46.8

9 Conclusions

Trajectory planning and obstacle avoidance problem for a robot manipulator is studied in this research. Four variants (PSO-C, PSO-W, qPSO-C and qPSO-W) of Particle Swarm Optimization (PSO) and Differential Evolution (DE) are proposed to solve this problem. Simulation experiments on a 5 degree-of-freedom (DOF) robot manipulator in an environment with static obstacles are conducted. Joint angles are used to generate the strings for DE and PSO. Mechanisms are introduced in these algorithms to escape from local optima during the search process. The performance of the four variants of PSO compared to the results reported in the literature. With the optimal parameters identified through sensitivity analysis, the four PSO variants are performing better than the earlier reported results. Performances of all five algorithms are evaluated. Since all the algorithms converge to a satisfactory error margin, convergence time and loop count are used as the parameters for comparison. Based on the results presented, it is concluded that qPSO-C is performing better in free space while PSO-C is performing better when obstacles are present. The performance of DE improves with the increase in number of obstacles. Additional experiments are to be conducted to test the performance of DE. Future work is also planned to conduct experiments in 3D real environments with feedback control to minimize the error.

References

1. Kim, S.W., Lee, J.J.: Resolved motion rate control of redundant robots using an adaptive fuzzy logic. Second IEEE International Conference on Fuzzy Systems, pp. 333–338(1993)
2. Beheshti, M.T.H., Tehrani, A.K.: Obstacle avoidance for kinematically redundant robots using an adaptive fuzzy logic algorithm. In: Proceedings of the American Control Conference, vol. 2, pp. 1371–1375 (1999)
3. Nearchou, A.C.: Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. Mech. Mach. Theory **33**(3), 273–292 (1998)
4. Secară, C., Vlădăreanu, L.: Iterative genetic algorithm based strategy for obstacles avoidance of a redundant manipulator. In: Proceedings of American Conference on Applied Mathematics, Stevens Point, pp. 361–366, USA (2010)
5. Zhang, Y., Wang, J.: Obstacle avoidance for kinematically redundant manipulators using a dual neural network. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **34**(1), 752–759 (2004)
6. Price, K., Storn, R.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **II**, 341–359 (1997)

7. Liu, Yu., Ni, F.-l., Liu, H., Wen-fu, X.: Enhancing pose accuracy of space robot by improved differential evolution. J. Cent. South Univ. **19**, 933–943 (2012)
8. Saravanan, R., Ramabalan, S., Balamurugan, C.: Evolutionary collision-free optimal trajectory planning. Int. J. Adv. Manuf. Technol. **36**, 1234–1251 (2008)
9. Gang, H., Li, D., Yang, J.: A research on particle swarm optimization and its application in robot manipulators. In: Pacific-Asia Workshop on Computational Intelligence and Industrial Application, PACIIA, vol. 2, pp. 377–381 (2008)
10. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, vol. 1, pp. 39–43 (1995)
11. Wen, X., Sheng, D., Huang, J.: A Hybrid Particle Swarm Optimization for Manipulator Inverse Kinematics Control. In: Huang, D.-S., Wunsch II, D.C., Levine, D.S., Jo, K.-H. (eds.) ICIC 2008. LNCS, vol. 5226, pp. 784–791. Springer, Heidelberg (2008)
12. Donelan, P.S.: Singularities of robot manipulators. Singul. Theory, pp. 189–217 (2007)
13. Desa, S.M., Qussay, A.S.: Image subtraction for real time moving object extraction. In: International Conference on Computer Graphics, Imaging and Visualization, CGIV 2004. Proceedings, pp. 41–45 (2004)
14. Qidwai, U., Chi-hau, C.: Digital image processing: an algorithmic approach with MATLAB. Chapman & Hall/CRC, London (2009)
15. Chyan, G.S., Ponnambalam, S.G.: Obstacle avoidance control of redundant robots using variants of particle swarm optimization. Rob. Comput.-Integr. Manuf. **28**(2), 147–153 (2012)
16. Hartenberg, R.S., Denavit, J.: A kinematic notation for lower-pair mechanisms based on matrices. ASME Journal of Applied Mechanics **22**(77), 215–221 (1995)
17. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, vol. 1, pp. 39–43 (1995)