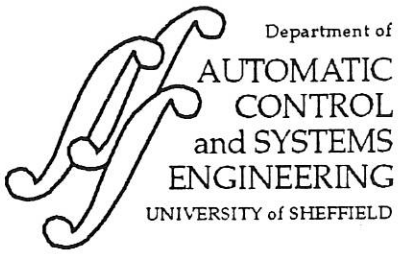
Monograph:
Rana, A.S. and Zalzala, A.M.S. (1997) Evolutionary Optimisation for Robot Motion. Research Report. ACSE Research Report 669 . Department of Automatic Control and Systems Engineering



EVOLUTIONARY OPTIMISATION FOR ROBOT MOTION

A.S. Rana and A.M.S. Zalzala

Robotics Research Group
Department of Automatic Control and Systems Engineering
The University of Sheffield
Mappin Street, Sheffield S1 3JD, United Kingdom

Research Report #669
February 1997

Tel : +44 (0)114 2225250
Fax : +44 (0)114 2731729
EMail : rrg@sheffield.ac.uk
Robotics Research Group

Evolutionary Optimisation for Robot Motion

A S Rana and A M S Zalzala

Robotics Research Group
Dept. of Automatic Control and Systems Engineering
University of Sheffield
Sheffield S1 3JD, UK
email: rrg@sheffield.ac.uk

Abstract

This paper presents a detailed analysis of a motion planner based on genetic algorithms for collision free motion planning of robotic manipulators through simulation. The problem is formulated for a 2-DOF planar manipulator moving in the presence of a static circular obstacle in its operational space. The algorithm is then extended to 3-DOF planar manipulator moving among multiple static obstacles.

Key words: Genetic algorithms, robotic manipulators, optimised collision free motion planning.

1. Introduction

Robotic manipulators are used in the industry to achieve more versatility than hard automation, which refers to the use of special purpose machines built to perform predetermined tasks. Once the robots are not needed for a particular task any longer, they can be reprogrammed and used for another task. Most of the tasks in industry are repetitive pick-and-place operations. At the present, in most of the cases the reprogramming of robots is done by the operator by guiding the robot through a sequence of points in its workspace and storing this sequence in the memory. However, for an intelligent robotic system, it is desirable that the robots work within a certain degree of autonomy without human supervision. For this purpose, an algorithm for evolutionary optimisation of robot paths is presented.

The planning of geometric path for a manipulator in the presence of obstacles in the environment is a search operation. Monte-Carlo techniques have proved to be very powerful methods in searching for good solutions in large, complex search spaces, such as the motion planning of a high degree-of-freedom (DOF) robotic manipulator moving in an environment filled with obstacles (Barraquand and Latombe, 1990). Genetic Algorithms (GA's) are also a guided random search technique, and they show better results than conventional hill-climbing methods when applied to this problem (Chen and Zalzala, 1995; Cleghorn et al., 1988).

Generations of trajectories for robotic manipulators is an order dependent process. Therefore, the representation of the path of the robot using GA's is slightly different from conventional coding to preserve the order of the process.

The details of the neural networks-based collision detection engine and evolutionary algorithm being analysed in this paper can be found in Rana and Zalzala (1995, 1996). These details will not be repeated here, but a brief discussion of the evolutionary algorithm will be given only.

2. Problem Formulation

The path planning is carried out in configuration space of the manipulator. The entire path of the robot is considered simultaneously as a string of via-points $\{\mathbf{p}_0,\mathbf{p}_1,\ldots,\mathbf{p}_i,\ldots,\mathbf{p}_N\}$ joining the initial configuration $\mathbf{p}_0$ and the final configuration $\mathbf{p}_N$, where $\mathbf{p}_i$ is the ith via-point given by the ordered pair $(\theta_{1i},\theta_{2i})$ and N is the total number of via-points on the path. These via-points are then fitted with parametric cubic-splines. Repeated modification is carried out to the position of the via-points through an evolutionary search to find a collision free path.

3. Collision Detection

In order to determine collision between the manipulator and the static obstacles, both the manipulator and the obstacles are approximated with touching circles (Beaumont and Crowder, 1989). The distance between the centres of these circles and the static obstacles is checked. If this distance happens to be less than the sum of the radii of the circles, the manipulator is colliding with the obstacle. Otherwise, it lies in the free space. An alternative to this is to use neural networks for collision detection (Rana and Zalzala, 1996). The input to the neural networks is the joint-angles of the manipulator, and the output varies between 0 and 1, depending upon whether the manipulator lies in free space or is colliding with the obstacle.

4. Evolutionary Program for Motion Planning of a 2-DOF Planar Manipulator

The details of this algorithm can be found in Rana and Zalzala (1995). However, a brief discussion is given for the benefit of the reader.

4.1 Encoding of Paths as Strings

For N via-points, the paths are encoded directly as chromosomes of the evolutionary program as

$$\mathbf{p}_1 : \mathbf{p}_2 : \cdots : \mathbf{p}_{i-1} : \mathbf{p}_i : \mathbf{p}_{i+1} : \cdots : \mathbf{p}_{N-1} \quad (1)$$

where $\mathbf{p}_i$ is the vector forming the ith gene in the chromosome, and represents the ith via-point on the path in the configuration space of the manipulator, and : is the concatenation operator. Thus, each gene in the chromosome consists of a vector with floating point components.

4.2 Fitness Function

Different objectives to be minimised by the evolutionary algorithm are

C_1 = penalty on the length of the path in configuration space.

C_2 = Penalty on the uneven distribution of via-points on the path.

C_3 = Penalty on collision with the obstacles.

The penalty on collision between the manipulator and the obstacles is given by

$$C_3 = \max_i (K_i);\ i=1,2,..,(N\text{-}1); \quad (2)$$

where

$$K_i = \begin{cases} 1 & \text{if the manipulator collides with the obstacle in the } i\text{th configuration} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

This function is referred to as *hard-threshold collision function*. The reason for this is that it only tells whether the manipulator is colliding with the obstacle or not, and does not give any information as to how far the manipulator is from the obstacle. The value of the function steps from 0 to 1 when the manipulator moves from free space into the c-obstacle without a gradual slope. If neural networks are used to map the collision between the manipulator and the obstacle in configuration space, they provide not only the information about whether the manipulator is colliding with the obstacle or not, but also give a gradual slope when the manipulator moves from the c-obstacles to free c-space. If this component of the entire objective function is generated by using the neural networks, it is referred to as *neural networks collision function*, and is given by

$$C_3 = \sum_i^{N-1} K_i \quad (4)$$

where K_i is the output of the neural networks for ith via-point.

The fitness function is formulated in two ways:

(i) Linear combination of objectives

(ii) Prioritization of objectives

(i) Linear Combination of Objectives

The objective function which is to be maximised is given by a linear combination of objectives as

$$g = k_1 \frac{C_1}{C_{1average}} + k_2 \frac{C_2}{C_{2average}} + k_3 \frac{C_3}{C_{3average}} \quad (5)$$

where k_1, k_2 and k_3 are positive constants and $C_{1average}$, $C_{2average}$ and $C_{3average}$ are the averages of the penalties on path length, the uneven distribution of via-point on the path and collision with the obstacles, respectively, calculated for the initial population at the beginning of the search. The fitness function is again formulated in two ways. In the first case, the fitness function is formulated as

$$F = \begin{cases} C_{\max} - g & \text{if } g < C_{\max} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where C_{max}, is a positive constant. It may be pointed out that if the hard threshold collision detection is chosen, then the maximum value of C_2 would be 1, so the value of $C_{2average}$ is taken to be equal to 1 in order to give it the same weight as the other objectives. The normalisation of all the penalties by dividing them by their respective initial averages provides with a way to visualise the weight associated with each penalty easily, thus making it easy to choose the values of k_1, k_2 and k_3. The value of k_3 is kept relatively higher, since collisions are to be avoided at all costs. The value of k_2 on the other hand is kept very low, since the via-points are redistributed evenly by one of the operators (the redistribution operator) in the algorithm. The value of C_{max} is chosen so that it is higher than the expected value of g at all times.

One problem associate with the selection based on this type of fitness function is that a string with a relatively higher fitness could entirely fill up the entire population very quickly, thus resulting in a premature convergence of the algorithm to a sub-optimal solution. To counter this problem, a *ranking* of the population has been investigated, in which the strings in the population are not chosen according to their fitness value, but according

to their rank among the population. This rank of an individual depends upon its fitness in a descending order in the population. A linear ranking has been used, in which the rank varies from a maximum value to a minimum value linearly (Chipperfield et al., 1994).

(ii) Prioritization of Objectives

Among the three objectives defined which are to be minimised by the algorithm, the collision avoidance is a constraint, whereas the other two objectives, i.e. penalty on the path length and penalty on the uneven distribution of via-points are the objectives which are to be minimised. One way to handle these two different types of objectives is to formulate a fitness function which depends on the priority of the objectives. Highest priority is assigned to the collision avoidance, and minimisation is performed first on this objective. The other objectives are minimised at a lower priority. The fitness function is defined as

$$g = \max\left(\left\| k_1\left(\frac{C_1 - C_{st}}{C_{1average} - C_{st}}\right) + k_2\left(\frac{C_2}{C_{2average}}\right)\right\|, \left\| k_3\left(\frac{C_3}{C_{3average}}\right)\right\|\right) \tag{7}$$

Where C_{st} is the length of straight line path in configuration space between the initial configuration and the final configuration. Fitness function F is defined by (6). The value of k_3 is kept higher than that of both k_2 and k_1 to give it a higher priority.

4.3 Generation of Initial Population

The initial population is generated at random for the evolutionary algorithm to work with. This is done by fitting Bezier curves between the initial and the final configuration at random, and then distributing the via-points over these curves at equal distance.

4.4 The GA Operators

The following four operators are used in the evolutionary algorithm:

a. Reproduction: The strings are reproduced for the next generation based on their fitness function. A weighted roulette wheel is used to select the strings from an old population for a new population.

b. Cross-over: The individuals in the population reproduced from the old population based on their fitness are grouped at random into pairs of parent strings. Same cross-over site is chosen at random among two parent strings. A cross-over is then performed by switching position of the via-points between this site among the two parent strings to produce two off-spring strings. This operation is carried out with a certain probability and only if the distance between the cross-over sites is less than a certain value.

c. Redistribution: The via-points are fitted with parametric cubic splines and then redistributed over these splines at equal distance to make the distribution even.

d. Relaxation: The path is then made to behave like a stretched string and relax under the strain.

e. Mutation: In order to carry out the mutation (which is done at a probability of *mutation_probatility*) any gene (via-point) in the chromosome is selected, and random values within a specific range are added to all the components in the gene.

f. Regeneration: New trajectories are generated and are injected into the population after every generation.

5. Simulation Results

Different simulations were carried out for the path planning algorithm. Detailed tests were performed to analyse the performance of the algorithm for the simple case of a single 2-DOF planar manipulator first, and effect of variation of different parameters in the evolutionary program were observed. Then the simulations to the extended case of a 3-DOF planar manipulator were performed. This section gives the results of these simulations.

5.1 An Analysis of Results to Evaluate the Performance and Working of the Evolutionary Program

It is possible in conventional GA's to explain their working through a 'schemata theorem'. But this is only possible in very simple cases. If the GA becomes complex, its working cannot be explained in terms of the schemata theorem. For evolutionary program presented here, which does not work in a binary space but directly in the problem space of the manipulator, this theorem cannot be applied at all. Hence the only way to evaluate the performance of the program is through experimental analysis. This section presents the results of the experimental tests to which the evolutionary program has been subjected.

The problem is checked for a single 2-DOF manipulator in the presence of a static circular obstacle first. The length of each link is considered to be 2 units, with the base of the robot located at (3.0, 0). The centre of the circular obstacle with a radius of 0.25 units is located at (3.0, 2.75).

Table 1. Parameters for the evolutionary program to plan motion of 2-DOF planar manipulator.

Population size	18
Number of via-points	30
Number of strings passed on to the next generation without cross-over	1
Number of new individuals generated during each generation	2
Number of iterations for which the redistribution/relaxation operator is applied	4
k_1 -Weight associated with penalty on length	1.5
k_2-Weight associated with penalty on uneven distribution of via-points over the path	1.0
k_3-Weight associated with penalty on collision	0.1
C_{max}-Constant used in linear combination of the objectives	5.0
Cross-over distance	1000.00
Cross-over probability	1.0
Mutation Probability	0.01

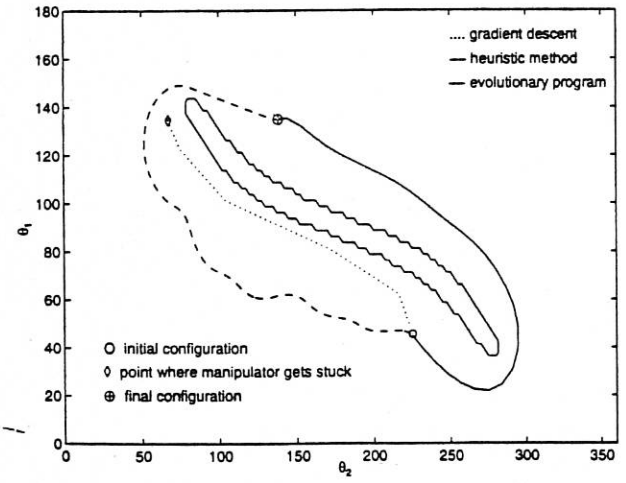


Fig. 2. Paths of the manipulator planned in configuration space by using hill-climbing technique, heuristic method and evolutionary algorithm.

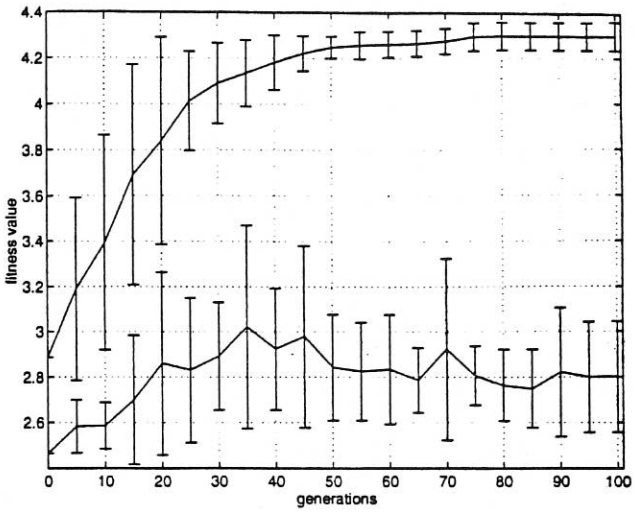


Fig. 1. Profile of best fitness and average fitness for the collision free path of a single 2-DOF manipulator in the presence of a static circular obstacle.

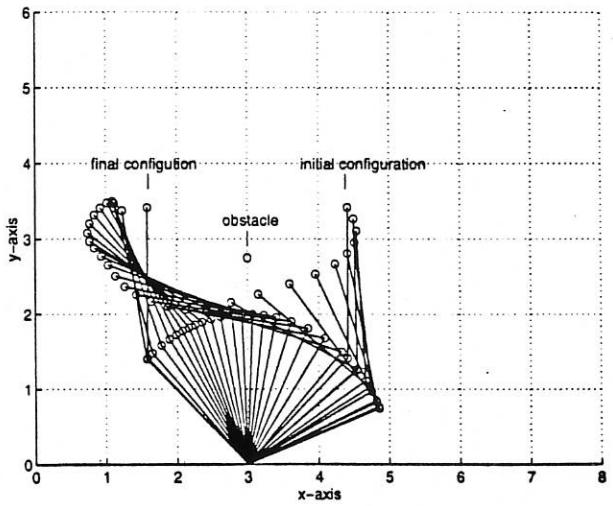


Fig. 3. Path of the manipulator in operational space planned by using the evolutionary algorithm.

Performance and Working of the Evolutionary Program

The evolutionary program is examined to find out whether it works for different random walks for a set of parameter values given in Table 1. Figure 1 shows the best fitness and average fitness for the same problem (for the path planned between the initial configuration of (45 °,225°) and final configuration of (135°,135°) in configuration space) for 20 different random walks. It has been observed that the standard deviation is less than 5.6% of the total change in the value of fitness function, indicating that the experiments are repeatable. Thus it can be said that the algorithm is able to find out the solution successfully.

Figure 2 gives the planned paths in configuration space for the same problem by using the hill-climbing technique, the heuristic method (Rana and Zalzala, 1996) and the evolutionary program for the sake of comparison. It can be observed that the hill-climbing method fails to find a solution. The length of the path planned by using the heuristic method is 302.9° and the length of the path planned by using the evolutionary program is 281.74°, since the path planned by the

evolutionary algorithms is optimised in terms of path length. Figure 3 gives the path of the manipulator in operational space.

Experiments for Parameter Optimisation

The parameter set given in Table 1 is chosen and variation in the value of different parameters is made. The following tests have been carried out:

TEST #1: Different formulations of fitness function and the effect of variation in different parameters in the fitness function.

Two different formulations for the fitness function as described in section 4.2 were tried out to check the performance of the path planning algorithm. Both the ranked population and un-ranked population was considered in linear combinations of the objectives as well as the prioritazation of the objectives. The results of the simulations are shown in Table 2. The problem chosen was the same as that given in the previous section. The results represent the average taken over twenty trials for different random walks. They show the final fitness function after 100 generations, the standard deviation and the number of generations it takes to get to 90% of the value of fitness to which the algorithm finally converges. The objective representing the penalty on collision with the obstacles is formulated in two different ways, and the true fitness values used in the algorithm cannot be compared with each other under different formulations. Instead, a performance measure is formulated. Its value can go up to 100, and represents a linear combination of the path length and the uneven distribution of the via-points over the path.

$$performance = 100 - \left(k_1 \left(\frac{C_1}{C_{1average}} \right) + k_2 \left(\frac{C_2}{C_{2average}} \right) \right) \times 100 \quad (8)$$

TEST #2: Variation in population size

The effect of variation of number of individuals in the population was considered. The results have been tabulated in Table 3.

TEST #3: Variation in the number of new trajectories generated

The third test that was carried out analysed the effect of variation of number of new trajectories imported into the population. The simulation results are given in Table 4

TEST #4: Number of cross-over points

Effect of the number of cross-over sites during each cross-over operation was considered. Three different cases were considered. In the first case, only a single cross-over point during a cross-over operation was considered. In the second case, two cross-over points were considered. In the third case, the number of cross-over sites was not fixed, but it could vary randomly from one to a maximum of $N-1$, where N is the number of via-points in the string. The results after 150 generations are given in Table 5.

TEST #5: Variation in the cross-over distance

Variation in the cross-over distance was considered in

Table 2. Results of simulations for different combinations of objective functions

	Linear combination of the objectives				Prioritization of the objectives			
	Hard threshold		Neural networks		Hard threshold		Neural networks	
	unranked	ranked	unranked	ranked	unranked	ranked	unranked	ranked
Fitness	29.87	31.52	30.47	30.71	31.01	31.16	32.79	31.20
Standard deviation	5.96	3.64	3.20	1.79	4.09	2.66	1.91	2.51
Convergence to 90% of final value	37	28	32	30	28	22	21	23

Table 3. Results of simulations with a variation in the number of individuals in the population.

Population Size	4	6	8	10	12	18	24	30	40	50	76
Fitness	4.052	4.266	4.272	4.340	4.295	4.294	4.308	4.344	4.377	4.382	4.399
Standard Deviation	0.119	0.084	0.084	0.062	0.072	0.073	0.042	0.020	0.031	0.018	0.024
Convergence to 90% of the final value	48	26	35	22	26	37	30	32	25	27	36

Table 4. Results of simulations with a variation in the number of new trajectories generated in each generation.

Number of new trajectories generated	2	4	6	8	12	18
Fitness	4.294	4.257	4.233	4.186	4.087	4.106
Standard Deviation	0.073	0.057	0.079	0.108	0.166	0.070
Convergence to 90% of the final value	37	35	24	30	30	31

the fifth test. The results showed that the only difference that increasing the cross-over difference had was that the algorithm converged to the final value earlier (40 generations for a cross-over distance of 1000 as compared to 65 generations when this distance is 10). There is no significant difference in the final value of the fitness reached or the standard deviation from the average value.

TEST #6: Variation in the cross-over and mutation probability

Variation in the cross-over probability and the mutation probability was considered in the sixth test. The results indicated that if the cross-over probability was decreased, there was no effect on the final fitness, but the number of generations it took the algorithm to reach the final value was increased with a decrease in this probability (35 generations with a probability of 1.0 as compared to 45 generations with a probability of 0.5). The mutation probability did not have any considerable effect on the performance of the algorithm at all.

5.2 Simulation Results for 3-DOF Planar Manipulator with Five Static Obstacles in the Workspace

Finally, the behaviour of the algorithm was checked when extended to 3-DOF planar case with five obstacles in the workspace. Each link of the manipulator was considered to be of unit length. The rectangular obstacles were enclosed by circles, and the collision between the

Table 6. Parameters for the Evolutionary Program for a 3-DOF planar manipulator moving in the presence of static obstacles.

k_1	1
k_2	0.25
k_3	1.0
C_{max}	6
population size	100
new trajectories	2
keep best	1
cross over distance	unlimited
cross over probability	1.0
mutation probability	0.01

Table 5. Simulation results for variation of number of cross-over sites.

	Single cross-over	Double cross-over	Multiple cross-over
Fitness function	4.344	4.264	4.283
Standard deviation	0.020	0.042	0.056
Convergence to 90% of final value	32	44	46

obstacles and the manipulator was checked by determining the minimum distance between the centre of the circles and the links of the manipulators. Different parameters of the evolutionary program are given in Table 6. Figure 4 shows the motion of the manipulator in which the path planning algorithm fails to find a collision free path when the penalty on the uneven

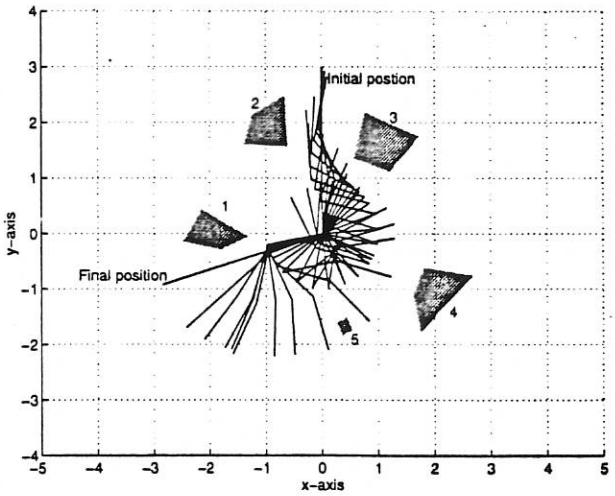


Fig. 4. A case showing the failure to plan collision free path when penalty on uneven distribution of via-points on the path is not considered.

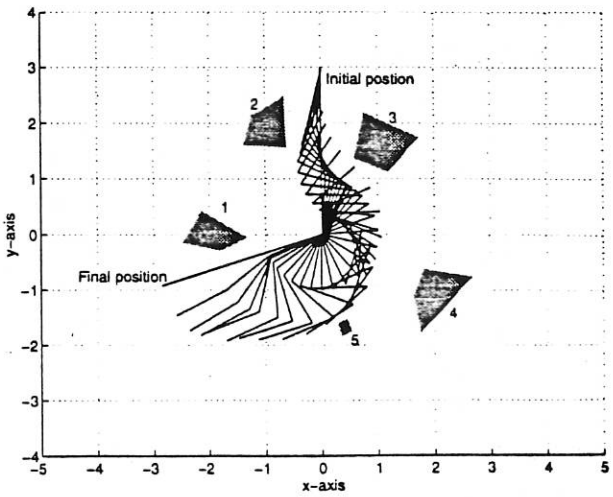


Fig. 5. A collision free path for a 3-DOF planar manipulator moving in the presence of multiple static obstacles in the operational space.

distribution of the paths is neglected. This case occurs in one in twenty runs of the evolutionary program. Figure 5 shows a sample collision free path of the manipulator in operational space when this penalty is taken into account. The success rate in this case is 100%. This compares to a success rate of only 60% for twenty trial for a similar evolutionary algorithm (Doyle, 1995).

5.3 Analysis of the Results

The simulation results highlight the effect of variation of different parameters of the algorithm. It is indicated that formulation of optimisation problem subject to constraints as Pareto-based optimisation (Fonseca and Fleming, 1995) using neural networks shows better results than that in which it is posed as a non-Pareto based optimisation, i.e. when hard-threshold collision detection function is used. Moreover, ranking of the population gives better results. Prioritization of the objectives also gives better results than that in the case of linear combination of the objectives, but ranking the population when neural networks are used actually decrease the efficiency of the algorithm. The variation in population size indicates that even though better results are achieved by increasing the population size, the algorithm works quite well with even very small population sizes. Increase in the number of new trajectories imported into the population after every generation decreases the value of final fitness achieved, but not using this operator altogether decrease the flexibility of the algorithm, and hence it does not show good results for small population sizes. By limiting the cross-over distance, the convergence to the final value becomes slower, but the effect on the final value of fitness achieved is not significant. The mutation rate does not have any effect on the algorithm at all, showing that this operator is not needed, since the importing of new trajectories into the population fulfils the purpose of this operator.

6. Conclusions

The analysis of an evolutionary algorithm for optimised motion planning of robot manipulator is presented. It is indicated that even though the variation in different parameters of the algorithm have some effect on the efficiency, this is not significant. This shows that the algorithm is quite robust to the variation in the values of these parameters. Problem-domain knowledge has been incorporated in the genetic-based search, and operators have been introduced (the redistribution and the relaxation operators) which do not directly correspond to any operator in natural evolution. The algorithm can work quite well with very low population sizes, and converges more quickly than conventional GA's. Hence it is computationally less intensive. It has also been indicated that this algorithm can be extended to more complex cases by extending it to the case of a 3-DOF planar manipulator moving amongst multiple static obstacles.

References

Barraquand J. and J.C. Latombe (1990). A Monte Carlo algorithm for path planning with many degrees of freedom. *IEEE International Conference on Robotics and Automation (Cincinnati, OH, USA)*, pp. 1712-1717.

Chen M and A.M.S. Zalzala, (1995). A genetic approach to motion planning of redundant mobile manipulator systems considering safety and configuration. *Research Report No. 559*, Dept. of Automatic Control and Systems Engineering, University of Sheffield, UK.

Chipperfield A., P. Fleming, H. Pohlheim and C. Fonseca (1994). *Users Guide: Genetic Algorithm Toolbox--For Use with Matlab®* , Department of Automatic Control and Systems Engineering, University of Sheffield, UK.

Cleghorn T.F., P.T. Baffes and L. Wang (1988). Robot path planning using a genetic algorithm. *Second Annual Workshop on Space Operation and Robotics,* Vol. 3019, Ch. 44, pp. 383-389.

Doyle, A.B. (1995). *Algorithms and Computational Techniques for Robot Path Planning*. PhD Thesis, University of Wales, Bangor.

Fonseca C.M. and P.J. Fleming (1995). Multi-objective optimization and multiple constraint handling with evolutionary algorithms I: A unified formulation. *Research Report 564*, Department of Automatic Control and Systems Engineering, University of Sheffield, UK.

Rana, A. S. and Zalzala, A. M. S. "An Evolutionary Algorithm for Collision Free Motion Planning of Multi-arm Robots", 1*st IEE/ IEEE conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, UK, 7-9 Sept. 1995.

Rana, A. S. and Zalzala, A. M. S. "A Neural Network Based Collision Detection Engine for Multi-arm Robotic Systems", *Research Report No. 615*, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK, Feb. 1996.