Robotica (1997) volume 15, pp 213–224. © 1997 Cambridge University Press

A genetic path planning algorithm for redundant articulated robots

Andreas C. Nearchou and Nikos A. Aspragathos

Department of Mechanical Engineering, University of Patras, 26 100 Patras (Greece)
e-mail: nearchou@upatras.gr, asprag@mech.upatras.gr

(Received in Final Form: September 19, 1996)

SUMMARY

In some daily tasks, such as pick and place, the robot is requested to reach with its hand tip a desired target location while it is operating in its environment. Such tasks become more complex in environments cluttered with obstacles, since the constraint for collision-free movement must be also taken into account. This paper presents a new technique based on genetic algorithms (GAs) to solve the path planning problem of articulated redundant robot manipulators. The efficiency of the proposed GA is demonstrated through multiple experiments carried out on several robots with redundant degrees-of-freedom. Finally, the computational complexity of the proposed solution is estimated, in the worst case.

KEYWORDS: Genetic algorithms; Path planning; Obstacle avoidance; Redundant robots.

1. INTRODUCTION

Collision-avoidance is an absolutely essential requirement for a robot to complete a desired task in an environment with obstacles. In order to achieve such complex tasks robots should be redundant, i.e. should possess at least one degree-of-freedom (DOF) more than the number required for the general free positioning. Exploiting the redundant DOF results in greater dexterity and flexibility for the robot's motion so that very complicated tasks can be tackled.

There is a rather extended literature on the development of collision-free path planning algorithms. An early work[1] proposed the use of Ternary algebra as a method for detection of interference, among planar shapes or among prismatic bodies. A good review of the work on the geometric interference between objects is also presented[2,3] with algorithms to solve the collision avoidance. Lozano-Perez's approach is based on the characterization of the position and orientation of an object as a single point in its configuration space (C-space) by building geometric objects, called configuration space obstacles (C-obstacles). These algorithms have the advantage that the intersection of a point relative to a set of objects is easier to deal with than the intersection of objects among themselves.

Many approaches have been proposed by the research community to solve the path planning problem using C-space. The most extensively studied so far, reduces the problem to a shortest-path problem in a graph. The visibility graph approach[2,4] is based on the construction of an undirected graph whose vertices are the initial and the goal configuration of the robot and the vertices of the C-obstacles. The problem of path planning is thereby converted to a search of the graph for a path (usually the shortest) between the initial and the final configurations. The retraction method[5,6] uses a Voronoi diagram to solve the path planning problem. The edges of the Voronoi diagram represent paths that are equidistant from the closest pair of obstacles, and its vertices are points where three or more such paths meet. A solution to the problem is found by searching this graph for shortest path. Brooks[7] proposed the freeway method for an explicit representation of the free space, based on overlapping generalized cones having straight spines and non increasing radii. Translations are performed along freeways and rotations at the intersections of freeways. A similar method based on the concept of good representation of the free space and adaptable for articulated robots appeared in reference 8. This work determines a spine of the free space between the obstacles and proposes a kinematics control algorithm which guides the tip of the hand to a desired point of the free space, while the manipulator's body is kept close to the spine of the free space. Khatib[9] introduced the potential field approach. This approach treats the robot as a point moving in its C-space under the influence of an artificial potential field produced by the final desired configuration, and the C-obstacles. Actually, the idea with this approach is that the final configuration produces an ''attractive'' potential which pulses the robot towards its goal, and the C-obstacles generate a ''repulsive'' potential which pushes the robot away from them. The negated gradient of the summation of these potentials is treated as an artificial force applied to the robot in order to control its motion. The decomposition of the free-space of the robot into cells is another important approach to solve path planning problem. The methods based on this approach include the Quad Tree and the Oct Tree methods.[10,11] Quad and Oct Trees are hierarchical data structures that recursively subdivide the work space of the robot into cells until a certain criterion is satisfied. In the case of the Quad Tree the workspace is represented by a rectangle and in each iteration this rectangle is subdivided into four smaller rectangles; where in the case of the Oct Trees the workspace is

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

represented by a cube which recursively is subdivided into eight smaller cubes (called octants). Again, a solution can be found by searching the trees for a valid path.

With the exception of the potential field approach, few from the above approaches have been successfully applied on articulated robots with redundant DOF. The main reason is that the construction of the C-obstacles for articulated robots, i.e. the representation of the robot as a point and the required mapping of the physical obstacles into the robot's configuration space, is a very difficult and time consuming task to be developed.[12–14] In spite of their effectiveness, potential field methods due to the use of fast descent techniques for optimizing the artificial potential function may get stuck into local optima.

Minimum distances between objects is another important aspect for collision avoidance. Several path planning algorithms which require the computation of the minimum distances between the robot and the obstacles have been appeared in the literature.[15–17] The main idea with this approach is to use the minimum distance measure to determine if any components of the pair of objects are about to collide.

In a previous work[18] the authors of this paper presented a technique to detect collisions between articulated robots, and planar convex obstacles. The proposed technique decomposes the original problem into two subproblems: a problem of determining the robot's links too closely located to the obstacles so that they are candidate to collide, and a problem of checking for interference between the links in question and the associated nearby obstacles. A collision between a link and an obstacle is detected by first building the convex hull of the points corresponding to the position of the two joints of the link and the vertices of the polygonal obstacle, and then, by searching this hull to locate the positions corresponding to the two joints. A collision occurs if one of the following exists: (a) none of the joints is contained in the hull, (b) only one joint is in the hull and the other lies inside the polygon, (c) the two joints are contained in the hull in not successive locations.

Recently, a number of researchers have been experimenting with genetic algorithms (GAs) for the path planning problem. Davidor[19] used a GA with dynamic chromosomes' structures and a modified crossover operator to optimize robot trajectories in environments free of obstacles. Assuming a predefined map consisting of knot points Shibata and Fukuta[20] and Shing and Parker[21] proposed genetic algorithms techniques to solve path planning problem. Using dynamic chromosomes and a set of special genetic operators Lin et al.[22] proposed an evolutionary algorithm for the path planning problem in mobile robot environment, which may contain unknown obstacles. The main characteristic of the above genetic approaches is that they presuppose the representation of the robot as a single point moving among its C-obstacles, therefore it is very difficult to be applied on articulated manipulators with redundant DOF.

This paper presents a new GA solution to the path planning problem of articulated redundant robot manipulators. The proposed solution maintains the robot in its physical Cartesian space and thus it does not require the computation of the robot's C-obstacles. The collisions avoidance between the robot's links and the obstacles is achieved by always keeping the links in a safe distance from the obstacles. The search problem is formulated as an optimization problem with objective the minimization of the Euclidean distance between the current and the final desired position/orientation of the robot's end-effector, satisfying the constraints related to the collision-free motion of the robot.

There are several features of GAs that make them attractive for use in this problem. GAs are theoretically and empirically proven[23] to provide a robust search in complex spaces with discontinuities. They are able to reach a global optimal solution in a large complex search space which can be multimodal and nonlinear. The constraints related to the obstacles in the robot's environment require the search of such complex spaces. Furthermore, the GAs have no requirements of any special characteristics of the problem's search space such as continuity of derivatives, so virtually any objective function can be chosen for optimizing. Using GAs there is no need to compute the Jacobian matrix, so that any problem related to the inversion of this matrix is overcome. The proposed GA solution needs only the forward kinematics equations of the robot and the geometry of the obstacles in the robot's environment; characteristics which are provided through modern CAD systems.

The rest of the paper is organized as follows: Section 2 states the problem. Section 3 presents the proposed genetic path planning algorithm (GPPA) solution, while section 4 analyzes the obstacle avoidance schemes used by the algorithm. Section 5 demonstrates and discusses the efficiency of the proposed GPPA through multiple experiments applied on two planar manipulators with five and seven-DOF, respectively, and a spatial 7-DOF Puma-like manipulator. Section 6, examines the population by population behavior of the algorithm, while section 7 analyzes the computational complexity of the algorithm in the worst case. Finally, section 8 summarizes the contribution of the paper.

2. PROBLEM STATEMENT

Given the initial configuration of a robot manipulator, the desired final position/orientation of its end-effector, and the geometry of the obstacles in the robot's environment, the problem is to find a continuous collision-free movement for the robot which moves its end-effector from the initial position/orientation to the final desired.

The robot is either a redundant or a non-redundant articulated manipulator with revolute joints, and is the only moving object in its environment. Ignoring the dynamic properties of the robot, the robot's motion is only constrained within its kinematic model and the

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

obstacles in its workspace. The obstacles are assumed convex polygons in the plane, and convex polyhedral in the three-dimensional space with a fixed and known geometry. The boundary of a non-convex polygon/polyhedron is represented as a union of convex polygons/polyhedral.

A polygon is represented by a sequence of Cartesian coordinates (X, Y) (given in counter-clockwise manner) which correspond to its vertices. For example, the sequence $(v_1, v_2, \ldots, v_M, v_1)$ describes a polygon with M-vertices in total. Similarly, a polyhedron is represented through the set of its faces together with a sequence of Cartesian coordinates (X, Y, Z) corresponding to the vertices of each face. For example, if $(f_1, f_2, \ldots, f_k)$ is the set of the faces of a polyhedron, then the jth face is represented by the sequence $(v_{j1}, v_{j2}, \ldots, v_{jM}, j_{j1})$.

Therefore, in the two-dimensional space the set of obstacles is described by the relation:

$$Polygons = \{\mathrm{Poly}_i(\mathrm{Vertex}_{ij}(X, Y)): 1 \le i \le M\mathrm{Poly}, 1 \le j \le M\mathrm{Vertex}_i + 1\} \quad (1)$$

Where,

Poly_i	The ith polygon in the robot's environment.
Vertex_{ij}	The jth vertex of the ith polygon.
MPoly	The total number of polgyons in the robot's environment.
$M\mathrm{Vertex}_i$	The number of vertices of the ith polygon.

For three-dimensional space, obstacles are described using the following relation:

$$Polyhedra = \{\mathrm{Poly}_i(\mathrm{Face}_{ij}(\mathrm{Vertex}_{ijk}(X, Y, Z))): 1 \le i \le M\mathrm{Poly}, 1 \le j \le M\mathrm{Face}_i, 1 \le k \le M\mathrm{Vertex}_{ij} + 1\} \quad (2)$$

Where,

Poly_i	The ith polyhedron in the robot's environment.
Face_{ij}	The jth face of the ith polyhedron.
Vertex_{ijk}	The kth vertex of the jth face of the ith polyhedron.
MPoly	The total number of polyhedral in the robot's environment.
$M\mathrm{Face}_i$	The number of faces of the ith polyhedron.
$M\mathrm{Vertex}_{ij}$	The number of vertices of the jth face of the ith polyhedron.

Assuming that the robot's end-effector is initially at the position given by the vector $P_{\mathrm{init}} = [X_{\mathrm{init}} \quad Y_{\mathrm{init}} \quad Z_{\mathrm{init}}]^{\mathrm{T}}$ and the desired goal position is given by $P_{\mathrm{goal}} = [X_{\mathrm{goal}} \quad Y_{\mathrm{goal}} \quad Z_{\mathrm{goal}}]^{\mathrm{T}}$, then the positional error is defined as the Euclidean distance between P_{init} and P_{goal}:

$$\mathrm{Positional}_{\mathrm{Error}} = ((X_{\mathrm{init}} - X_{\mathrm{goal}})^2 + (Y_{\mathrm{init}} - Y_{\mathrm{goal}})^2 + (Z_{\mathrm{init}} - Z_{\mathrm{goal}})^2)^{1/2} \quad (3)$$

2.1 Problem formulation

This paper formulates the path planning problem as a constrained optimization problem. The objective of this optimization is to minimize the positional error given by the equation (3) subject to the constraints related to the obstacle avoidance. Thus, the problem is formulated as follows:

$$\begin{cases} \text{minimize } \mathrm{Positional}_{\mathrm{Error}} \\ \text{subject to Collision-free movement} \end{cases} \quad (4)$$

3. THE GENETIC PATH PLANNING ALGORITHM

To solve the problem given by equation (4) a genetic algorithm (GA) was used. GAs are powerful domain-independent search mechanisms which emulate the process of genetic evolution found in nature as a means of progressing towards the optimum.[24] Combining an artificial survival of the fittest with special genetic operators, GAs provide a robust search mechanism that is suitable for a variety of search problems.

3.1 A brief overview

The most frequently used form of GAs in a variety of engineering problems is the simple GA. A simple GA requires the natural parameter set of the real optimization problem to be coded as a finite-length string of bits. It first generates an initial population of such strings, and working iteration by iteration, generates new populations of strings until the convergence to an optimum solution. A new generation of strings is being produced by applying on the entire population three artificial genetic operators: reproduction, crossover, and mutation. Each one of these operators emulates a corresponding process found in biological evolution.

Reproduction is a process where an old string is carried into a new population according to its fitness value; where fitness is defined as a non negative merit being maximized. In other words, during this process each string is evaluated and those with the highest scores are selected to be reproduced, i.e. to get copies of their structures in the next generation. The traditional method implementing reproduction is the roulette wheel selection.[23] According to this method the probability to select a string for reproduction is given by the relation: $P(\mathrm{select}) = f_i / \sum_j^n fj$, where f_i the fitness value of the ith string in the population, and n the total number of strings in the population.

Crossover follows reproduction and is a recombination operator that works on a pair of old strings randomly selected according to a specified crossover probability. It actually performs a 'cut-paste-and-patch' operation. In a simple crossover, a cross site is chosen at random along the string length and then the positions are swapped among the two strings following the cross site. As an example, consider two strings S_1 and S_2 of length 7 mated at random from the mating pool of the new generation:

$$S_1 = 100 \mid 0110$$
$$S_2 = 011 \mid 0001$$

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

Assuming that the cross site chosen at random is the 3rd position, the resulting crossover yields a pair of new strings (offsprings) S_1'', S_2'':

$$S_1'' = 100 \quad 0001$$
$$S_2'' = 011 \quad 0110$$

Mutation is simply an occasional random alteration on a string position based on a specified mutation probability. Since a strong position corresponds to a digital value of '1' or '0', mutation results to a change of '1' to '0' and vice versa.

3.2 Why go genetics?

There are at least five basic reasons that make GAs well suited for use in this problem: First, they provide a robust search in large and complex spaces finding nearly global optima. This is very important for our problem because the constraints related to the obstacles in the robot's workspace generate a search space with discontinuities and non-linearities, and the robot's DOF makes this space large enough. Therefore, classical optimization methods (e.g. hill-climing methods) depending upon restrictive requirements of continuity and derivative existence, and due to their inherently local scope of search are unsuitable for the solution of this problem. Second, GAs do not require any form of smoothness although the search space contains discontinuities and non-linearities. Third, taking into account the size of the search space, as well as their ability to reach a global optima. GAs are relatively fast, especially when tuned to the domain on which they are operating. Fourth, they do not need the computation of the Jacobian matrix so that any problem related to the inversion of this matrix like singularities is overcome. Fifth, tuning a GA for a particular domain is relatively easy; it only needs the specification of the following characteristics:

- A representation mechanism, i.e. a way of encoding problem's solutions to artificial chromosomes.
- A way of initializing the population of chromosomes.
- An evaluation mechanism, i.e. the computation of a function called fitness function for each chromosome.
- The application of special genetic operators (reproduction, crossover, mutation) on the population in order to generate new populations with better chromosomes.
- Values to some control parameters e.g. population size, crossover rate and mutation rate.

3.3 Chromosome's syntax

The critical aspect in designing a GA is the basic mechanism that links the GA to the real problem which has to be solved. This mechanism is twofold: firstly a way of *encoding* solutions to the real problem on artificial chromosomes (representation mechanism), and secondly an *evaluation* of a function (fitness) that returns a measure of how good an encoding is (evaluation mechanism). The chromosome selected for use in this work is an m-bit string with the following syntax: the first m/DOF-bits correspond to the first joint of the robot, the next m/DOF-bits correspond to the second joint, etc. Therefore, each joint angle can be calculated from the value of its m/DOF-bit string using the mapping shown below:

$$\Theta_i = \Theta_i^{\min} + \left(\frac{\text{Bitvalue}}{{}_2(m/DOF)_{-1}}\right) * (\Theta_i^{max} - \Theta_i^{\min}) \qquad (5)$$

where, Θ_i is the variable for the ith joint, and $\Theta_i^{\min}$, $\Theta_i^{\max}$ the corresponding minimum and maximum limits (depending on the robot's geometry). Bitvalue is the decimal value of the chromosome's bits corresponding to the ith joint.

3.4 Fitness function evaluation

The fitness function, also known as the scoring function of a chromosome's solution, corresponds to the objective function of the constrained optimization problem we want to minimize.

GAs are essentially unconstrained search procedures within the given representation space. The traditional GA formulation for constrained optimization problems is through the use of penalty function.[23] However, as Davis discusses in reference 25, though the evaluation function may be well defined, there is no accepted methodology for combining it with the penalty. In this work, to overcome the above problem, the following function formulation is used:

$$\text{fitness} = \begin{cases} (1 + \text{Positional}_{\text{Error}})^{-1}, & \text{If collision-free movement} \\ 0, & \text{Otherwise} \end{cases} \qquad (6)$$

The objective of the proposed GA is to maximize the above fitness function. This results to a minimum value for the $\text{Positional}_{\text{Error}}$. In the ideal case where fitness takes the value of one, the corresponding generated $\text{Positional}_{\text{Error}}$ takes the smallest possible value, i.e. a zero value. Chromosome's structures which violate the constraint of the collision-free movement, i.e. represent not feasible robot configurations, take a small fitness value (actually a zero value), and therefore a very small chance to survive and reproduce in the next generations. Following Darwin's principle of natural selection, the formulation of equation (6) progressively results in the predominance of feasible solutions in the population.

Figure 1 displays the flow-chart of the proposed GPPA. After the termination of the algorithm a sequence of successive, feasible robot configurations is returned.

4. OBSTACLE AVOIDANCE SCHEMES

In this work, the collision-free movement of the robot is achieved by always keeping the robot's links in a safe distance from the physical obstacles. To do so, during the robot's motion the proposed GPPA computes the distance between each link and each obstacle. A joint motion is acceptable if this distance is greater than a user-defined lower limit.

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

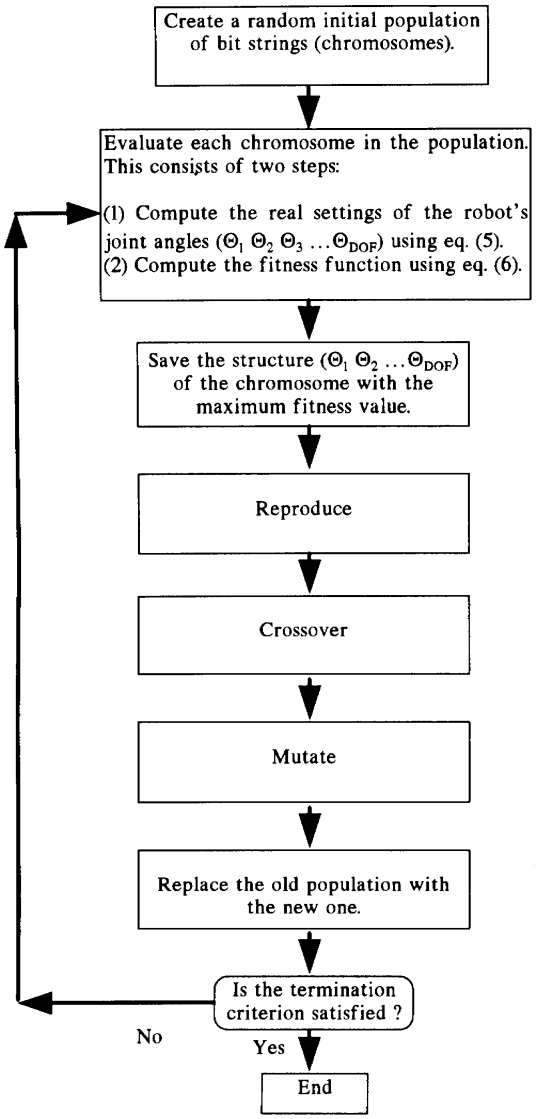


Fig. 1. The flow-chart of the proposed GPPA.

4.1 Obstacle avoidance in the plane

It is assumed that the links of the robot are straight linear segments with end-points the corresponding pair of joints. Therefore, the obstacle avoidance scheme consists of computing the safe distance between each straight linear segment and each convex polygon. This computation is given in algorithmic form below:

For the ith link of the robot-arm, with $i \in [1 \cdots \text{DOF}]$ (Figure 2):

a) Assure that both joints of the link lie outside the polygonal obstacle. If this is not true, then we have a collision.
b) Find the minimum distance between the two joints of the link and the vertices of the polygon a_i.
c) Find the minimum feasible distance between the two joints of the link and the polygon's edges (b_i). A distance b_i is feasible if the foot-point of the normal to a specific polygon's edge lies between the end-points of the edge and outside the polygon.
d) Find the minimum feasible distance between the polygon's vertices and the robot's link (c_i). A distance c_i is feasible if the foot-point of the normal to the link lies between its two joints.

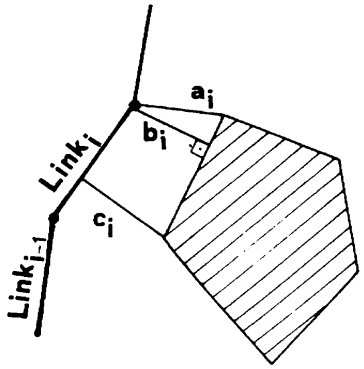


Fig. 2. Computing the safe distance between a robot's link and a polygonal obstacle.

e) Determine the minimum between the measures a_i, b_i, c_i.
f) If this measure is equal or greater than a user-defined lower-bound, then the link does not collide with the obstacle, and this measure is the safe distance between the link and the obstacle. Otherwise a collision occurs.

The above computations must be carried out between each link of the robot and every one of the obstacles. The larger the number of the links and the obstacles in the robot's workspace, the more the time required to complete these computations. In several cases, a link is "far" enough from an obstacle and thus checking for possible interference is superfluous. To reduce the time complexity of these computations, the following simple technique is suggested:

a) Enclose each one of the robot's links and every obstacle in a circle. A link's circle has radius equal to the half of the link's length. An obstacle's circle corresponds to the minimum spanning circle[26] which encloses the polygonal obstacle.
b) Compute the Euclidean distance between the center of a link's circle and an obstacle's circle.
c) If this distance is greater than the summation of the radii of the two circles, then the link is guarantee in a safe distance from the obstacle. In this case, we say that the link is "far" from the obstacle.
d) Else the link is "near" the obstacle and there is a possibility for collision which can be detected by the proposed scheme shown in Figure 2.

An aspect of the above complexity-reduction mechanism is demonstrated in Figure 3 for three successive links

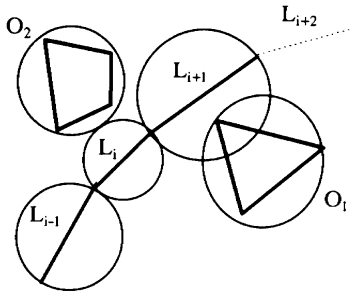


Fig. 3. Determining the links too "close" to obstacles.

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

L_{i-1}, L_i, L_{i+1} of a planar robot moving among two polygonal obstacles O_1, O_2. As we can see from this figure, only link L_{i+1} of the robot is "near" an obstacle, actually near the obstacle O_1 and therefore, the collision detection mechanism must be applied only for this link.

4.2 Obstacle avoidance in the three-dimensional space

It is clear that obstacle avoidance in the three-dimensional space is more difficult than that in the plane. In general, the computational complexity of the problem is a function of two parameters:[27] the number of the algebraic constraints which define the free space of the robot, and the number of the robot's DOF.

Using an analogous obstacle avoidance scheme the proposed GPPA can be applied in three-dimensional problems too. For simplicity, it is assumed that each link of the robot is enclosed in a cylinder of radius r_i ($i = 1 \cdots \text{DOF}$), and each obstacle O_j ($j = 1 \cdots M\text{Poly}$) is bounded by a convex polyhedron. A polyhedron's representation is given by the equation (2). It is also supposed that the orientation of the robot's hand is not given and thus it is not accounted in the following. The representation of a link by a cylinder and not by a polyhedron although reduces the robot's flexibility it is computational profitable. Further, the links of a redundant robot is in general longish and thus the cylinder's representation is convenient.

Let the ith link (L_i) of the robot is surrounded by a cylinder. The axis of the cylinder is defined by the common normal to the rotation axes of the i and $i+1$ joints. r_i is the minimum ray of all the cylinders with the above defined axis which surrounds link L_i. Let L_i be rotated with an angle $\Delta\Theta_i$ about the rotation axis of the i-joint (Figure 4). This rotation results to a change in the position of the $(i+1)$-joint. The possible collisions after this motion between L_i and the obstacle are:

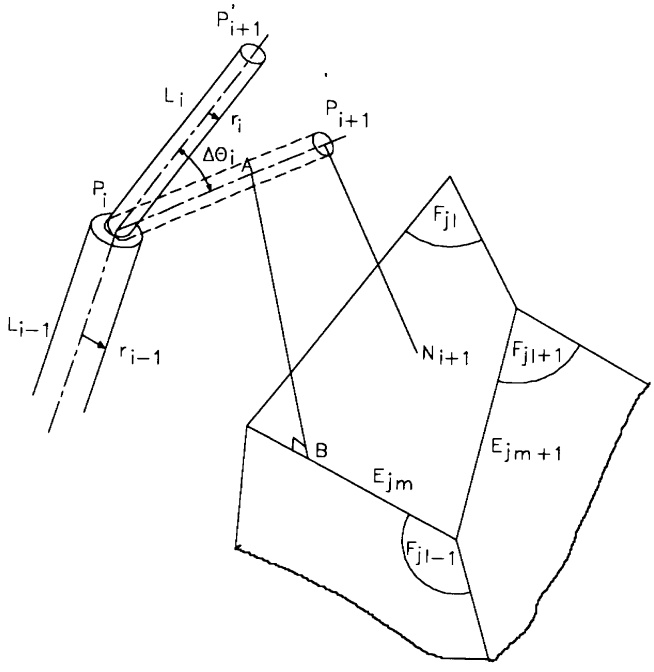


Fig. 4. Computation of a safe distance between a link and a polyhedral obstacle.

A) Collision with an obstacle's face.
B) Collision with an obstacle's edge.
C) Collision with a vertex of the obstacle.

Suppose that the position Pi of the ith joint has been checked from the previous movement of the link L_{i+1} and is safe. Therefore, in the present step it must be checked: (a) the distance of the point P_{i+1} (corresponding to the position of the $i+1$ joint) from each face F_{lj} ($l = 1 \cdots M\text{Poly}$, $j = 1 \cdots M\text{Face}_1$), (b) the magnitude of the common normal between the segment P_iP_{i+1} and each edge E_{lk} ($k = 1 \cdots M\text{edges}_1$). Where, according to equation (2) F_{1j} denotes the jth face of the 1-obstacle in the robot's workspace, and $M\text{edges}_1$ the total number of edges of the 1th obstacle. To accomplish these checks the proposed obstacle avoidance scheme executes the following steps:

A) Collision with an obstacle's face: Find the point N_{i+1} of the intersection of the normal from point (P_{i+1} to the plane of the face F_{lj}, by using the well known relations from the analytic geometry. Check if N_{i+1} belongs to the face or lies outside of it by using the ray casting technique.[28] If N_{i+1} is into the face and the magnitude of the straight linear segment $P_{i+1}N_{i+1}$ is less than or equal to the radius r_i, then we have a collision between L_i and the obstacle and thus this motion is prohibited.

B) Collision with an obstacle's edges: The previous checks will be failed when the magnitudes of the normal from points P_i and P_{i+1} to the neighborhood face e.g. F_{ji} and F_{ji+1} (Figure 4), respectively, pass the above tests, however the common edge E_{jm+1} of these faces intersects the cylinder L_i.

Due to this case, the proposed scheme finds the common normal AB between the segment P_iP_{i+1} and each edge E_{im}. If the magnitude of the common normal AB is less than r_i and the points of intersection A and B lie into the segment P_iP_{i+1} and E_{im}, respectively, then a collision occurs.

C) Collision with an obstacle's vertex: To check for a collision between the link and a vertex of the obstacle we must determine if point B lies inside the edge E_{im} extended by the radius r_i to each side. If this happened, then we have a link-vertex collision.

5. EXPERIMENTAL RESULTS AND DISCUSSION

The efficiency of the GPPA was estimated for multiple experiments carried out on two planar manipulators with five and seven-DOF, respectively, and for a spatial Puma-like robot with seven-DOF. The algorithm was implemented in Pascal programming language and tested on an IBM-486/66 machine.

A significant task in implementing a GA is the selection of the suitable settings for the algorithm's control parameters. In this work, Grefenstette's parameters,[29] were selected for use: i.e. a population of 30 strings, crossover and mutation rates equal to 0.95 and 0.01, respectively. The evaluations were done for a maximum of 50 generations. Furthermore, the elitist selection strategy for the chromosomes' reproduction was used and a linear scaling for the fitness function.

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

The GPPA was defined to terminate when one of the following conditions occur: (a) the minimum positional error in the current generation is less than a lower user-defined threshold, (b) the population's average fitness value exceeds the 0.95 of the maximum fitness.

Using the mapping given by equation (5), each joint angle of the robots is represented by a 10-bit string, and each such string is concatenated to produce a string representing a unique robot-configuration. Therefore, a 50-bit string was used for the five-DOF robot and a 70-bit string for the seven-DOF robots, respectively.

In the first experiment (Figure 5) the five-DOF robot must place its end-effector with free orientation on specific locations corresponding to some "knot" points on a straight line parallel to X-axis. The robot's motion is constrained by a rectangular obstacle shown in Figure 5 by the hatched region. The robot has links of the same length and equal to 50 cm and a fixed base on the Cartesian position (0, 0) cm. The robot's joint angles can take values in the range [−45, +225] degrees. Figure 5 illustrates the results of the numerical simulation of the GPPA for these experiments. As we can see from this figure, the robot avoids the obstacle and places its end-effector with an acceptable accuracy on the desired locations on the dashed lines. Table I summarizes in the first two columns the Cartesian coordinates of the desired "knot" points. The last column of the table displays the corresponding final positional errors produced by the GPPA.

Further experiments (3,000 in total) applied on the five-DOF robot produced the results shown in Figure 6. In each one of these experiments, the robot was ordered to pose its end-effector at a different Cartesian position randomly selected inside its free space. the GPPA was first tested for the case where no obstacles are in the robot's environment, and then for the case where the rectangular obstacle displayed in Figure 5, constrained its motion. Figure 6 displays the convergence probability of the proposed GPPA into a solution having final

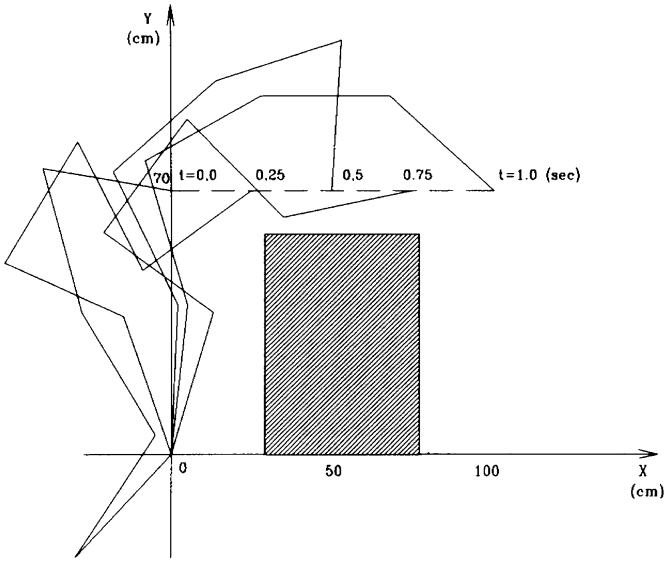


Fig. 5. Simulation results for the five-DOF robot in an environment with obstacle.

Table I. End-effector's positional errors generated by the proposed GPPA.

X (cm)	Y (cm)	Positional error (cm) experiment without obstacle	Positional error (cm) experiment with obstacle
0	70	0.027	0.071
25	70	0.048	0.095
50	70	0.009	0.193
75	70	0.160	0.181
100	70	0.083	0.174

positional error less than x mm. The convergence is plotted in percentage rates.

As we can see from Figure 6, in environment free of obstacles the GPPA converged to an error less than 0.5 mm in the 40% of the experiments, to an error less than 1 mm in the 72% of the experiments, and to an error less than 3 mm in the 93% of the experiments. Similarly, in the case with the obstacle in the robot's environment, the GPPA converged to the above errors nearly in the 22%, 50%, and 72%, respectively, of the experiments. Detailed examination of the desired positions for which the proposed GPPA produced a large error, showed that nearly 38% of them were lying at the boundary of the robot's workspace; a fact that explains the inability of the GPPA to reach them with a better accuracy.

A more complex experiment is illustrated in Figure 7. In this experiment, a seven-DOF planar robot must move through a narrow passage generated by two polygonal obstacles (shown by the hatch lines) placing its end-effector with free orientation on a specific target location. The links of the robot are 50 cm in length and its joint angles can take values in the range [−15, 220] degrees. The achievement of the above requirements is clearly shown in Figure 7 by four different robot configurations generated by the algorithm.

Figure 8 shows the minimum positional error (maximum fitness) generated in each generation of the GPPA for the seven-DOF robot. The final minimum error in positioning generated by the algorithm was equal to 1.58 mm.

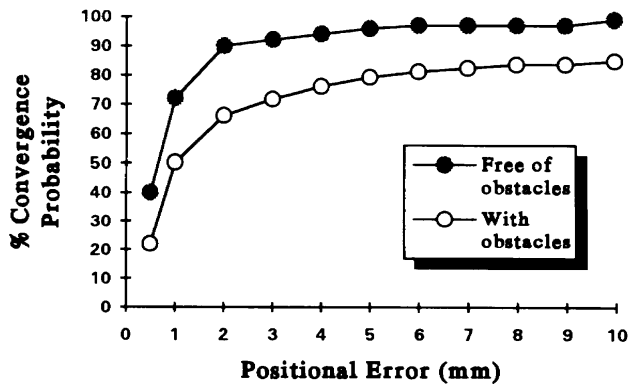


Fig. 6. Convergence probability vs. positional error for the planar five-DOF robot.

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

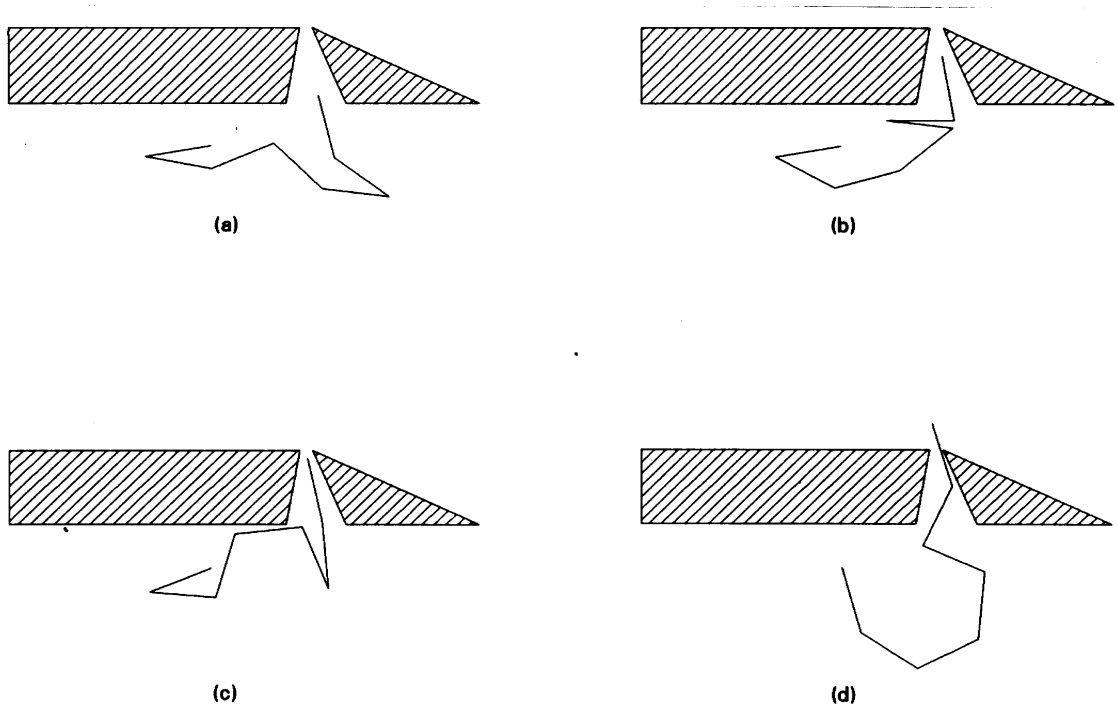


Fig. 7. The generated motion of a seven-DOF robot inside a narrow passage using the GPPA.

Finally, the efficiency of the proposed GPPA was estimated for problems in the three-dimensional space. To that purpose, a Puma-like robot (Figure 9) was selected for use. The robot has seven-DOF which allows the arbitrary positioning of its end-effector within the three-dimensional workspace.

Assuming that the motion of the robot is constrained by a rectangular cube shown in Figure 10, the robot is demanded to pose its end-effector on three different points (randomly selected inside its workspace) starting each time from the initial configuration. The desired goal points marked in Figure 10 as P1, P2 and P3, have Cartesian coordinates (expressed in cm): $[-43, 29, 70]$, $[-36, 40, 70]$, $[-55, 54, 0]$, respectively. The robot's links have the following lengths $a2 = 80$ cm, $a3 = 50$ cm, $d5 = 80$ cm, $d7 = 50$ cm. The robot's base is fixed at the Cartesian coordinates $(0, 0, 0)$ cm and its joint angles can take values in the following ranges: the first three angles in the range $[-125, 45]$ degrees, and the rest joints in the range $[-30, 225]$ degrees. Furthermore, it is assumed that the links of the robot are surrounded by cylinders of zero radius, $r_i = 0$ $(i = [1 \cdots 4])$.

Figure 11 demonstrates the results after the application of the proposed GPPA for the goal points P1, P2 and P3. Figure 11(a) shows the best robot configuration generated by the algorithm for the goal point P1 under two different view-angles. Figure 11(b) and 11(c) show the best configuration generated for the goal points P2 and P3, respectively. Table II, summarizes the minimum positional errors generated by the algorithm for the above three experiments.

6. EXAMINING THE BEHAVIOR OF THE GPPA. HOW WELL DESIGNED?

How well the GPPA was designed in order to converge into an acceptable solution? The current section deals with the answer to this question. If the GA has been correctly designed, then the population will evolve over successive generations so that the fitness of the best and average individual in each generation increases towards

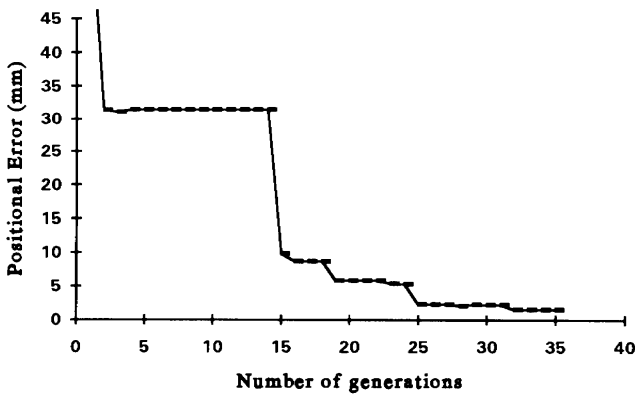


Fig. 8. The positional error generated by the GPPA for the planar seven-DOF robot.

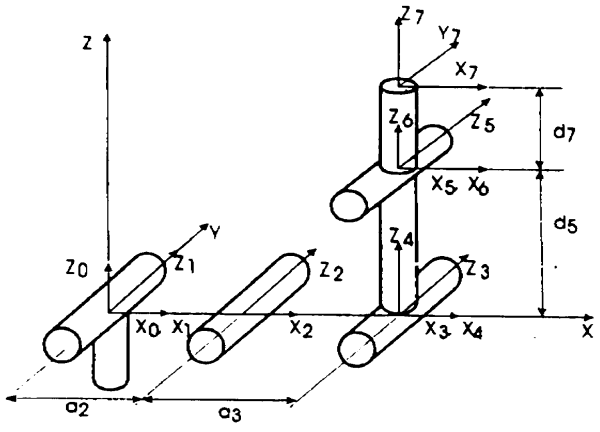


Fig. 9. The seven-DOF three-dimensional robot manipulator.

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

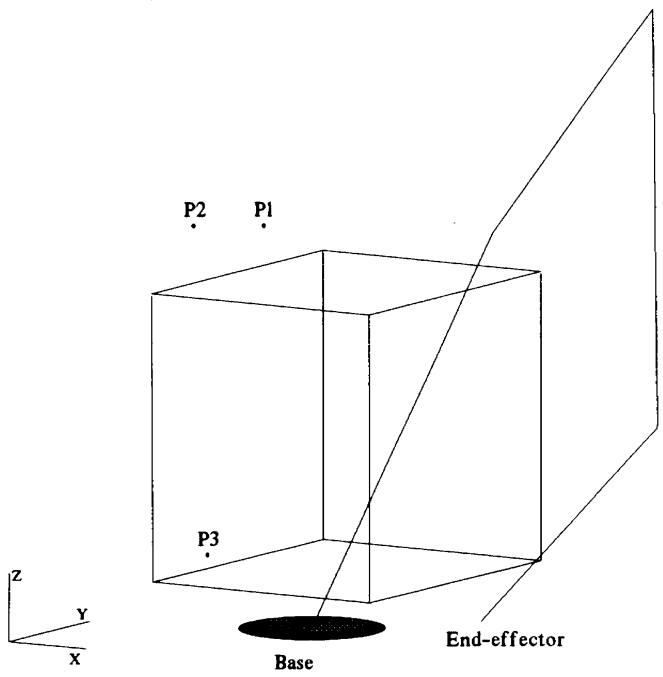


Fig. 10. The environment of the Puma-like seven-DOF robot.

Table II. Minimum positional error for the goal points P1, P2 and P3.

Point	Desired position			Generated position			Minimum positional error (cm)
	X	Y	Z	X	Y	Z	
P1	−43	29	70	−42.7	28.4	69.6	0.781
P2	−36	40	70	−35.7	40.1	69.2	0.860
P3	−55	54	0	−55.0	54.39	0.016	0.390

the global optimum. The process of convergence is equivalent to an increasingly uniform progress of the average fitness of the population until it reaches the 95% of the best fitness of the population.[30]

Figure 12 demonstrates the GA's behavior for the point-to-point motion of the seven-DOF planar robot. The results concern the experiment appeared in Figure 7. As we can see from this figure the average population's fitness is uniformly increased over successive generations until it reaches the 95% of the best (maximum) fitness of the entire population. The algorithm was finally

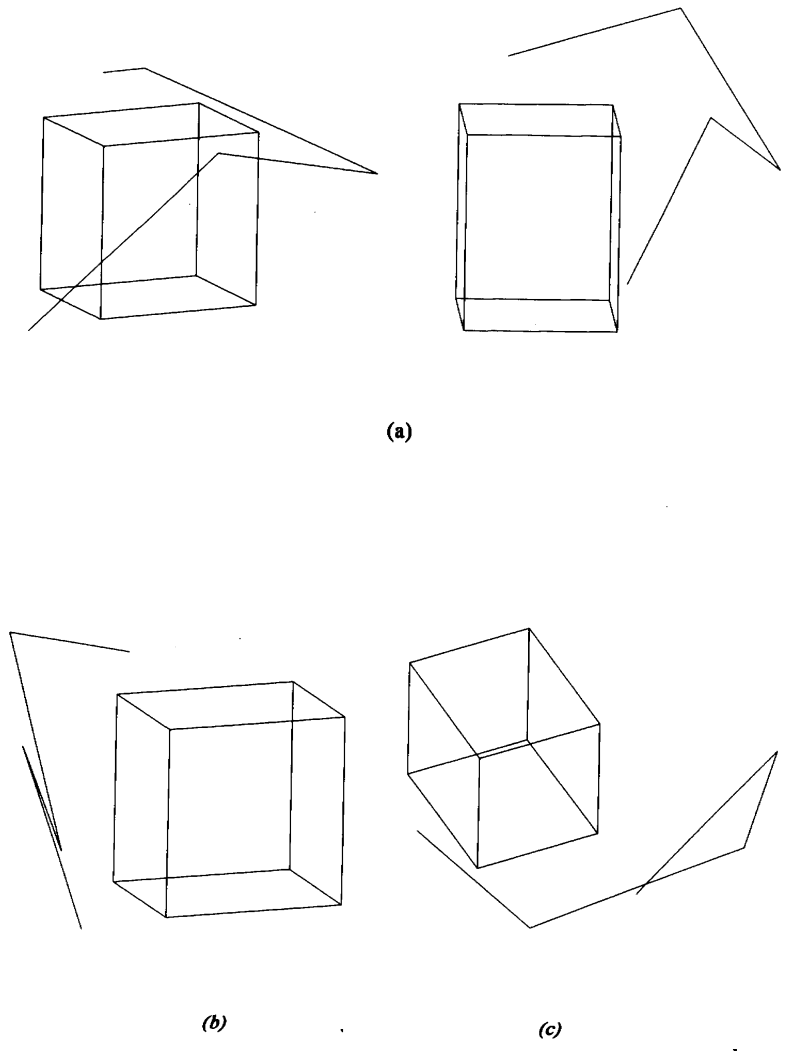


Fig. 11. Best robot's configurations for the goal points P1, P2 and P3.

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

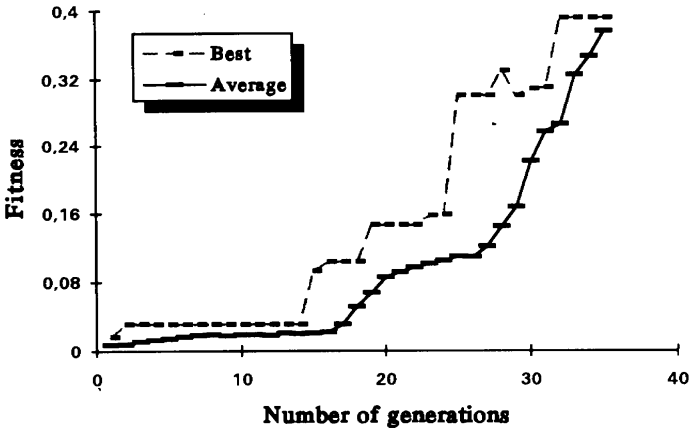


Fig. 12. Best (maximum) and average fitness over the successive generations of the GPPA for the seven-DOF planar robot.

converged in the 35th generation producing a minimal error in positioning equal to 1.58 mm (best fitness = 0.388).

A similar result of a typical application of the GPPA on the PUMA-like robot is displayed in Figure 13. The result concerns the movement of the robot to the goal point P1 (see Figure 10) in the absence of the polyhedral obstacle. We can see from Figure 13 that, as the value of the best population's fitness increases, an increasingly uniform progress of the population's average fitness observed. The GA reaches the global optimum when the value of the average fitness surpasses the 95% of the value of the best fitness of the entire population. Finally, GA produced a positional error equal to 1,534 mm (best fitness = 0·395).

The positional accuracy of the proposed algorithm is better, compared to that appeared in reference 31 although obstacles avoidance is also considered. In all the experiments the GPPA generates a collision-free robot configuration. Furthermore, the proposed algorithm is very simple and easy to handle by the operator. It only needs the kinematic model of the robot, the obstacles' geometry, and the desired goal end-effector's

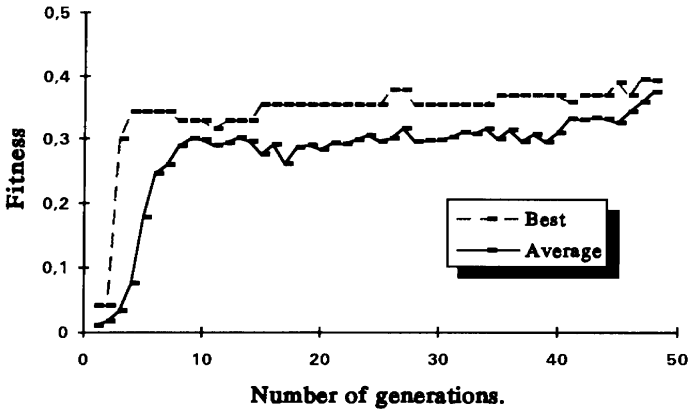


Fig. 13. Best (maximum) and average fitness over the successive generations of the GPPA for the Puma-like robot.

location, characteristics that are provided through the modern CAD systems. Any other intervention of the operator like the definition of the initial (start) configuration of the robot, the definition of a reference joint angle[32] or the estimation of special gain factors[33] is not required.

The basic properties of the GAs have been verified in all the experiments carried out in this work. More specific, although the search space is very complex, with discontinuities and non-linearities (due to the obstacles' appearance), GAs do not fall in local optima; instead, they generally find nearly global optima. This is very important comparing to traditional optimization techniques. For example, hill-climbing techniques are unsuitable for use since they get stuck in local optima. The use of potential function[9] has this disadvantage. Furthermore, GAs do not require any form of smoothness such as continuity of derivatives in order to work properly. This implies that virtually, ''any'' cost (or fitness) function can be selected for optimizing.

7. COMPUTATIONAL COMPLEXITY OF THE GPPA

The basic structure of the proposed GPPA was displayed in Figure 1. Let the population size of the GA is n, and the length of each chromosome is equal to m-bits. The GA can evolve in a maximum of G generations. It is assumed that a chromosome (a string of bits) is generated in constant time. Further, the generation of a random number in the range $[0 \cdots 1]$ needs constant $O(1)$ time. Examining the asymptotic time complexity of the GA in the worst case, the following bounds are found:

- The initialization process is executed only once, and consists of the random generation of n chromosomes for the initial population. With the above assumptions the execution of this process needs $O(n)$ time.
- The evaluation process in the worst case is executed G times, and its execution consists of two basic steps: (a) Convert the binary value of each chromosome to the corresponding decimal value. This computation needs $O(m)$ time. (b) Compute the fitness function of each chromosome. For problems free of obstacles, this step includes the computation of the Cartesian position of the robot's end-effector, and the computation of the Euclidean distance between this position and the final desired. The first computation is a function of the robot's DOF so it takes $O(DOF)$ time. The second computation needs constant $O(1)$ time. Therefore, the evaluation process needs for each chromosome $O(m + DOF)$ time. Since, $DOF \ll m$ (m is an integer multiple of the robot's DOF) then the required time is $O(m)$. Finally, for the population of all the chromosomes and until the termination of the algorithm, the process needs $O(n.m.G)$ time in the worst case.
- The reproduction process can be executed up to G times, and consists of the following: (a) Compute the total population's fitness $F = \sum_j^n f_j$, (b) for each

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

chromosome compute the selection probability (for reproduction) $p_i = f_i/F$, (c) for each chromosome compute the accumulative probability $cp_i = cp_{i-1} + p_i$ (with $i = 2 \cdots n$), (d) randomly generate a number in the range $[0 \cdots 1]$. The computations (a)–(d) are executed for each chromosome of the entire population and take constant time, therefore the process of reproduction costs $O(n.G)$ time in the worst case.

- The crossover process is executed in two steps: (a) Generate a random number in the range $[0 \cdots 1]$ for each chromosome, (b) check whether this number is smaller than the crossover probability p_c. These two steps take constant time, so that the crossover process requires $O(n.G)$ time in the worst case.
- In a similar way, the mutation process generates a random number in the range $[0 \cdots 1]$ for each bit in the population, and checks whether this number is smaller than the mutation probability p_m. These computations cost $O(n.m.G)$ time, in the worst case.

Putting together all the previous results leads to a worst case time complexity of $O(n.m.G)$. Recall, that this bound concerns the application of the proposed GPPA in environments free of obstacles.

In environments with obstacles the asymptotic behavior of the algorithm only differs in the evaluation process. The cost required for the collision-detection must also be included in the corresponding bound. This cost is spent during the computation of the fitness function of each chromosome, and depends on the obstacle avoidance scheme used by the GPPA. Here, we'll estimate only the complexity for the planar case. Thus, in the above results we must add the time needed to calculate the distance between a link and a nearby obstacle. Using the scheme presented in section 4.1, these computations take $O(k1)$ time, where k1 the number of vertices of the obstacle. If k the total number of vertices of all the obstacles in robot's environment, then in the worst case a specific link must be examined for possible collision with all the obstacles, which consequently yields an $O(k)$ time. This time must spent in the worst case for all the links of the robot, therefore a bound of $O(k.DOF)$ is received. Since, $DOF \ll m$ the previous bound can be written as $O(m.k)$. Thus, according to the previous analysis, evaluation process needs $O(n.m.k.G)$ time. This bound corresponds to the worst case time complexity of the algorithm in environments with obstacles.

In fact, the last result can be much smaller since: (a) A GA usually converges before the completion of all the G generations (see section 4). (b) With the mechanism presented in section 4.1 (Figure 3) only the links "too close" to an obstacle are examined for possible collision.

Further, the above bound can be drastically reduced, using a more effective obstacle avoidance scheme, e.g. by computing the minimum Euclidean distance between a link and an obstacle using the algorithm appeared in reference 34. This computation can be performed in optimum $O(\log k)$ time. This results to a worst case bound of $O(n.m.\log k.G)$.

In this paragraph the space complexity is analyzed. To store the n chromosomes of the entire population, and taking into account that a new population replaces the old one, the GA needs $O(n.m)$ space. This bound concerns problems without obstacles in the robot's environment. In problems with obstacles in the robot's environment one must add to the previous bound the space required to store the obstacles' vertices. Thus, the worst case of space complexity of the proposed GPPA is $O(n.m + k)$.

As a last comment, it must be emphasized that the number of bits selected for the chromosome's syntax is an integer multiplication of the robot's DOF. Therefore, according to the previous bounds, the time and space complexity of the proposed GA in the worst case, is a linear function of the robot's DOF. Furthermore, the number of bits chosen to represent each joint angle is depended on the required precision in number of decimal points of the generated solution. The 10-bits chosen for the representation of the real value of each joint angle in the above experiments, correspond to a precision of two decimal points, in average.

8. CONCLUSION

A method based on genetic algorithms was introduced to solve the problem of path planning of articulated redundant robot manipulators. The problem was formulated as a constrained optimization problem with the main objective of minimizing the end-effector's positional error, and a secondary objective of satisfying the constraints related to the collision-free movement of the robot. The proposed method maintains the robots in its physical Cartesian space and thus it does not need the construction of the configuration-space obstacles. The efficiency of the proposed algorithm was demonstrated through multiple experiments carried out on two planar robots with five and seven *DOF,* respectively, and on a three-dimensional PUMA-like manipulator with seven-*DOF.* Using techniques for safe distance computation, the algorithm guarantees the collision-free movement of the robot producing an acceptable end-effector's positional error. The algorithm runs in $O(n.m.k.G)$ time, and uses $O(n.m + k)$ space, in the worst case. Here n is the chromosome's length (an integer multiplication of the robot's DOF), m the population size, k the total number of vertices of all the obstacles in the robot's environment, and G is the maximum number of generations the *GA* can proceed.

REFERENCES

1. Tai-Soku and Bahram Revani, "Interference detection in computer-aided geometry design of prismatic objects" *ASME paper,* No. DET.84 (1979).
2. T. Lozano-Perez and M.A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles" *Communication of the ACM* **22,** No. 10, 560–570 (1979).
3. T. Lozano-Perez, "Spatial planning: a configuration space approach" *IEEE Trans. on Computers* **C-32,** No. 2, 108–120 (February, 1983).
4. H. Mitchell, "An algorithmic approach to some problems

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press

in terrain navigation'' *Artificial Intelligence* **37,** 171–201 (1988).
5. C. O'Dunlaing, M. Sharir and K. Yap, ''Retraction: A new approach to motion planning'' *Proc. of the 15th ACM Symposium on the Theory of Computing,* Boston (1983) pp. 207–220.
6. C. O'Dunlaing and K. Yap, ''A retraction method for planning the motion of a disc'' *J. Algorithms,* **6,** 104–111 (1985).
7. R.A. Brooks, ''Solving the find-path problem by good representation of free space'' *IEEE Trans. on Systems, Man and Cybernetics* **SMC-13**(3), 190–197 (1983).
8. M. Anthimopoulou and N. Aspragathos, ''Kinematic control of planar manipulators moving between obstacles'' *2nd International Workshop on Advances in Robot Kinematics,* Linz, (September, 1990) pp. 380–391.
9. O. Khatib, ''Real-time obstacle avoidance for manipulators and mobile robots'' *Int. J. Robotics Research* **5,** No. 1, 90–98 (Spring, 1986).
10. S. Arimoto and H. Noborio, ''A 3D closest pair algorithm and its applications to robot motion planning'' *IFAC Robot Control,* Karlsruhe, FRG (1988) pp. 471–479.
11. G. Garcia, Ph. Wenger and P. Chedmail, ''Computing moveability areas of a robot among obstacles using octrees'' *Proc. of the 4th International Conference on Advanced Robotics,* Columbus, Ohio, June 13–15 (1989) pp. 385–396.
12. Y.K. Hwang, ''Boundary equations of configuration obstacles for manipulators'' *Proc. of the IEEE Intern. Conf. on Robotics and Automation,* Cincinatti (1990) pp. 298–303.
13. Q.J. Ge and J.M. McCarthy, ''An algebraic formulation of configuration space obstacles for spatial robots'' *Proc. of the IEEE Intern. Conf. on Robotics and Automation,* Cincinnati (1990) pp. 1536–1541.
14. J.R. Dooley and J.M. McCarthy, ''Parameterized descriptions of the joint space obstacles for a 5R closed chain robot'' *Proc. of the IEEE Intern. Conf. on Robotics and Automation,* Cincinnati (1990) pp. 1536–1541.
15. E.G. Gilbert and D.W. Jonson, ''Distance functions and their application to robot path planning in the presence of obstacles'' *IEEE Journal of Robotics and Automation* **RA-1,** No. 1, 21–30 (1985).
16. G. Hurteau and N.F. Stewart, ''Distance calculation for imminent collision indication in a robot system simulation'' *Robotica* **6,** Part 1, 47–51 (1988).
17. J.E. Bobrow, ''A direct minimization approach for obtaining the distance between convex polyhedra'' *Int. J. Robotics Research* **8,** No. 3, 65–76 (1989).
18. A. Nearchou and N. Aspragathos, ''A collision-detection scheme based on convex-hulls concept for generating kinematically feasible robot trajectories. **In:** *Advances in Robot Kinematics and Computational Geometry* (J. Lenarcic and B. Ravani, eds., Kluwer Academic Publ. (1994) pp. 477–484.
19. Yuval Davidor, *Genetic Algorithms and Robotics: A heuristic strategy for optimization* (World Scientific Publishing, Singapore, 1991).
20. T. Shibata and T. Fukuda, ''Robot motion planning by genetic algorithm with fuzzy critic'' *Proc. 8th IEEE Int. Symp. on Intelligent Control* (August, 1993) pp. 565–570.
21. M.T. Shing and G.B. Parker, ''Genetic algorithms for the development of real-time multi-heuristic search strategies'' *Proc. 5th Int. Conf. on Genetic Algorithms* (Morgan Kaufmann Publ., Los Altos, California) (1993) pp. 565–570.
22. H.-S. Lin, J. Xiao and Z. Michalewitz, ''Evolutionary algorithm for path planning in mobile robot environment'' *Proc. IEEE Evolutionary Computation Conf.,* Orlando (1994) pp. 211–216.
23. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison Wesley Publishing Company, Reading, Mass., 1989).
24. J.H. Holland, *Adaptation in Natural and Artificial Systems*: An introductory analysis with applications to Biology, Control, and Artificial Intelligence (University of Michigan Press, Ann Arbor, 1975).
25. L. Davis, *Genetic Algorithms and Simulated Annealing* (Morgan Kaufmann Publishers, Los Altos, CA, 1987).
26. F.P. Preparata and M.I. Shamos, *Computational Geometry an Introduction* (Springer-Verlag, New York, 1985).
27. J.T. Schwartz and M. Sharir, ''A suvery of motion planning and related geometric algorithms'' *Artificial Intelligence J.* **37,** 157–169 (1988).
28. P.F. Rogers and J. Adams, *Mathematical Elements for Computer Graphics* (McGraw Hill, New York, 1976).
29. J.J. Grefenstette, ''Optimization of control parameters for genetic algorithms'' *IEEE Trans. on Systems, Man, and Cybernetics* **SMC-16,** No. 1, 122–128 (January/February, 1986).
30. K.A. De Jong, ''*An Analysis of the Behavior of a Class of Genetic Adaptive Systems*'' (Doctoral dissertation, University of Michigan) *Dissertation Abstracts International,* **36**(10), 5140B (1975).
31. J.K. Parker, A.R. Khoogar and D.E. Goldberg, ''Inverse kinematics of redundant robots using genetic algorithms'' *IEEE Int. Conf. on Robotics and Automation* (1989) **Vol. 1,** pp. 271–276.
32. Y. Nakamura, H. Hanafusa and T. Yoshikawa, ''Task-priority based redundancy control of robot manipulators'' *Int. J. Robotics Research* **6,** No. 2, 3–17 (1987).
33. A. Maciejewski and C. Klein, ''Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments'' *Int. J. Robotics Research* **4,** No. 3, 109–117 (1985).
34. F. Chin and C. Wang, ''Optimal algorithms for the intersection and the minimum distance problems between planar polygons'' *IEEE Transaction on Computers* **C-32,** No. 12, 1203–1207 (December, 1984).

https://doi.org/10.1017/S0263574797000234 Published online by Cambridge University Press