Cartesian path generation of robot manipulators using continuous genetic algorithms

Za’er S. Abo-Hammour a,∗, Nasir M. Mirza b,1, Sikander M. Mirza b,1, Muhammad Arif c,1

a Systems Engineering Department, Pakistan Institute of Engineering and Applied Sciences, P.O. Nilore, Islamabad 45650, Pakistan
b Department of Physics and Applied Mathematics, Pakistan Institute of Engineering and Applied Sciences, P.O. Nilore, Islamabad 45650, Pakistan
c Department of Informatics and Computer Science, Pakistan Institute of Engineering and Applied Sciences, P.O. Nilore, Islamabad 45650, Pakistan

Received 21 June 2001; received in revised form 11 April 2002

Communicated by F.C.A. Groen

Abstract

In this paper, the authors describe a novel technique based on continuous genetic algorithms (CGAs) to solve the path generation problem for robot manipulators. We consider the following scenario: given the desired Cartesian path of the end-effector of the manipulator in a free-of-obstacles workspace, off-line smooth geometric paths in the joint space of the manipulator are obtained. The inverse kinematics problem is formulated as an optimization problem based on the concept of the minimization of the accumulative path deviation and is then solved using CGAs where smooth curves are used for representing the required geometric paths in the joint space through out the evolution process. In general, CGA uses smooth operators and avoids sharp jumps in the parameter values. This novel approach possesses several distinct advantages: first, it can be applied to any general serial manipulator with positional degrees of freedom that might not have any derived closed-form solution for its inverse kinematics. Second, to the authors’ knowledge, it is the first singularity-free path generation algorithm that can be applied at the path update rate of the manipulator. Third, extremely high accuracy can be achieved along the generated path almost similar to analytical solutions, if available. Fourth, the proposed approach can be adopted to any general serial manipulator including both nonredundant and redundant systems. Fifth, when applied on parallel computers, the real time implementation is possible due to the implicit parallel nature of genetic algorithms. The generality and efficiency of the proposed algorithm are demonstrated through simulations that include 2R and 3R planar manipulators, PUMA manipulator, and a general 6R serial manipulator.

© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Robot manipulators; Singularity avoidance; Cartesian path generation; Inverse kinematics problem; Continuous genetic algorithms

1. Introduction

The impact of robots in industry is unquestionable. One aspect of interest in robotics is the optimum motion planning of the robotic continuous-path manufacturing processes in order to increase the
productivity and the precision of robot systems. The precision of the process, quality, may be improved at the programming level by defining the Cartesian path with more path points, employing a more sophisticated path generation method while the production rate, productivity, is improved by incorporating an efficient trajectory generation algorithm which results in the time optimal trajectories and correspondingly increases the production rate.

The path generation problem of continuous-path manufacturing process involves the use of the kinematics equations of the robot to obtain the joint angles at the given path points in Cartesian space; as a result, it is related to the inverse kinematics problem of manipulators that is a highly nonlinear problem. The solution strategies of the manipulator’s inverse kinematics problem are divided into two main classes; closed-form solutions and numerical solutions. The closed-form solution for the inverse kinematics problem is generally difficult to derive for general serial manipulators. In addition to that, only in special cases may robots with six degrees of freedom be solved analytically. For example, a sufficient condition that a manipulator with six revolute joints will have a closed-form solution is that three neighboring joint axes intersect at a point [9]. The redundant manipulators, on the other hand, have infinite number of solutions in joint space. As a result, an optimization method is required that tries to fully exploit the kinematic redundancy of the system in order to improve robot motion performance through the use of some criteria like obtaining smooth rather than fluctuating solutions in joint space, minimizing the displacements of the actuators, obtaining collision-free joint paths, etc.

A wide spectrum of numerical methods for the solution of the inverse kinematics problem of robot manipulators is reported in literature. One of the most powerful techniques used in this field is the genetic algorithm. The inverse kinematics problem is formulated as an optimization problem and is then solved using GAs that are based on the use of the forward kinematics model of the manipulator. As a result, they avoid the kinematic singularities of the manipulator. In this regard, Parker et al. [8] introduced genetic algorithms for solving the inverse kinematics problem of redundant manipulators where GAs were used to move a robot to a target location while minimizing the largest joint displacement from the initial position. In this technique, the accuracy of the final position is not ideal since it generates a large final positioning error. More recently, Davidor [23] proposed a special GA for path generation problem of redundant manipulators. He considered generating robot path as a typical order-dependent process and presented a GA model for this problem. The main characteristics of his algorithm are the use of dynamic individuals structures and a modified crossover operator called analogous crossover. The goal of the proposed GA is to minimize the accumulative deviation between the generated and the desired path. However, his proposed GA has drawbacks and could not fully exploit the abilities of GAs. First, due to the variable-length string used in path generation, the string length of an optimal solution is not known, i.e., the number of path points along the generated path in the optimal solution cannot be specified and is generally of small number of path points. Second, the proposed algorithm cannot be applied in applications where extremely high accuracy is required to meet high quality standards. It is basically an approximation method and fails to generate the desired path exactly due to the small number of path points used by the algorithm. The reader is referred to Davidor [3] for further details about the algorithm.

The motivation behind our work is to introduce a novel genetic algorithm called the continuous genetic algorithm (CGA) for the solution of the path generation problem, which has the following characteristics. First, it can be applied to any general serial manipulator with revolute and prismatic joints having positional degrees of freedom, which might not have a derived closed-form inverse kinematics solution. Second, the proposed approach guarantees the smoothness of the resulting joint paths in joint space since its operators are applied at the joint’s path level, i.e., of global nature, rather than the path point level. Third, it can be applied to both nonredundant and redundant manipulators while the previous applications of conventional GAs were limited only to redundant manipulators, which have infinite number of solutions in their solution space. Fourth, extremely high accuracy can be achieved along the generated path almost similar to analytical solutions, if available, while the precision of the generated paths using conventional GAs is unacceptable for applications where extremely high accuracy is required. This is due to the fact that CGAs can be applied at the path update rate while
conventional GAs use small number of path points, few tens, along the Cartesian path. Fifth, when combined with trajectory generation part, kinematic singularities are avoided since inverse kinematics equations are not used. Sixth, when applied on parallel computers, the real time implementation is possible due to the implicit parallel nature of genetic algorithms.

In addition to that, the applications of the proposed algorithm cover a very wide spectrum of optimization problems encountered in the motion planning of robot manipulators; it can be easily modified to handle the point-to-point motion planning where the continuous curves used in CGAs are applied to find the optimal solution curve for a given objective function, it can also be applied to collision-free motion planning by introducing the collision avoidance algorithm and modifying the objective function used to take this into account.

In this paper, we will introduce the continuous genetic algorithms to the reader and apply it to the path generation problem for both nonredundant and redundant manipulators in a free-of-obstacles workspace. The algorithm is based on the concept of the minimization of the accumulative path deviation only; no other objective functions are included in this work. The organization of the remainder of the paper is as follows: the formulation of the path generation problem for solution by the continuous genetic algorithms is described in Section 2. Section 3 covers the continuous genetic algorithms in detail. The effectiveness of the presented method is demonstrated in Section 4 through simulations of various types of manipulators. Finally, the conclusions are given in Section 5.

2. Formulation of the path generation problem

Let us consider a robot manipulator with $M$ degrees of freedom and $N$ task space coordinates. Assume that a desired Cartesian path, $P_{dc}$, is given, the problem is to find the set of joint paths, $P_{\theta}$, such that the accumulative deviation between the generated Cartesian path, $P_{gc}$, and the desired Cartesian path, $P_{dc}$, is minimum. In other words, we are interested in the determination of a set of feasible joint angles, which corresponds to a set of desired spatial coordinates of the end-effector in the task space.

In our approach, the desired geometric Cartesian path is uniformly sampled. The number of sampling points (path points or knots) is specified by the programmer and depends on the desired accuracy of the generated path. The accuracy of the generated path increases as the number of path points increases. However, a limiting case for this number is the path update rate; i.e., we can increase the number of path points till we reach the limit

$$N_k = T_t P_{ur},$$

where $N_k$ is the number of knots along the geometric path, $T_t$ is the total traveling time from the starting configuration to the final configuration along the desired path, and $P_{ur}$ is the path update rate of the manipulator.

The path update rate is normally within the range 20–200 Hz for most robotic applications [1]. As an example, the path update rate for PUMA 560 series robot is about 36 Hz [4]. This means that if the expected travelling time is 3 s, the number of knots along the desired path is 108 points. For a 20Hz path update rate, the number of points will be 60. Generally, the number of knots along the path should not exceed the $N_k$ value found from Eq. (1). If it exceeds that limit, then it will be of no benefit because the robot accuracy cannot exceed that limit at the running time.

It is to be noted that after the sampling process, $P_{dc}$ and $P_{gc}$ are matrices of dimension $N \times N_k$ while $P_{\theta}$ is a matrix of $M \times N_k$ dimension. After sampling the geometric path, at the path update rate for best accuracy, the generated values of the joint angles using the continuous genetic algorithms, $P_{\theta}$, are used by the direct (forward) kinematics model of the robot to obtain the generated Cartesian path given by

$$P_{gc} = F_k(P_{\theta}),$$

where $F_k$ represents the robot forward kinematics model.

The deviation between the desired Cartesian path, $P_{dc}$, and the generated Cartesian path, $P_{gc}$, at some general path point, $i$, is given as

$$E(i) = \sum_{k=1}^{N} |P_{dc}(k, i) - P_{gc}(k, i)|.$$

The accumulative deviation between the two paths (desired and generated) depends on whether the initial
and final joint angles corresponding to the initial and final configurations of the end-effector are given in advance using the inverse kinematics model of the manipulator or through other numerical technique (fixed end points) or the case in which the initial and final joint angles are not given (free end points). For the fixed end points case, the accumulative deviation between the two paths is given by the formula
\[ E = \sum_{i=2}^{N_k} \sum_{k=1}^{N} |P_{dc}(k,i) - P_{gc}(k,i)| = \sum_{i=2}^{N_k} E(i), \] (4)
while for the free end points case, the accumulative deviation between the two paths is given by the formula
\[ E = \sum_{i=1}^{N_k} \sum_{k=1}^{N} |P_{dc}(k,i) - P_{gc}(k,i)| = \sum_{i=1}^{N_k} E(i). \] (5)
The fitness function, a nonnegative measure of the quality of individuals, is defined as
\[ F = \frac{1}{1 + E}. \] (6)
The optimal solution of the problem is obtained when the deviation function, \( E \), approaches zero and correspondingly the fitness function, \( F \), approaches unity. As a result, the path generation problem is formulated as a minimization problem of the deviation function or as a maximization problem of the fitness function.

3. Continuous genetic algorithm

The genetic algorithm was first proposed by Holland [7], and since then GAs turned out to be powerful tools in the field of global optimization. This is undoubtedly due to its ability to solve multi-objective, nondifferentiable problems. The practical applications of GAs can be found in a number of engineering areas, including control, signal processing, computers, artificial intelligence etc.

GAs are based on principles inspired from the genetic and evolution mechanisms observed in natural systems. Their basic principle is the maintenance of a population of solutions to the problem that evolves in time. They are based on the triangle of genetic reproduction, evaluation and selection [5]. Genetic reproduction is performed by means of two basic genetic operators: crossover and mutation. Evaluation is performed by means of the fitness function that depends on the specific problem. Selection is the mechanism that selects parent individuals with probability proportional to their relative fitness.

The construction of a genetic algorithm for any problem can be separated in five distinct and yet related tasks [5]. First, the genetic representation of potential problem solutions. Second, a method for creating an initial population of solutions. Third, the design of the genetic operators. Fourth, the definition of the fitness function. Fifth, the setting of the system parameters, including the population size, probabilities with which genetic operators are applied, etc. Each of the above components greatly affects the solution obtained as well as the performance of the genetic algorithm.

The five mentioned factors resulted in the availability of numerous variants of GAs reported in literature. However, when using GAs in optimization problems, one should pay attention to whether there is some restriction on the smoothness of the resulting solution curve or not. In case that smoothness of the solution curve is not required then the conventional genetic algorithm (discrete version) will perform well. On the other hand, if smoothness of the solution curve is a must, then the CGA is preferable in this case. CGAs were used by Gutowski [6] to solve one-dimensional optimization problems. The CGA is that algorithm which depends on the evolution of curves in one-dimensional space. In general, CGAs use smooth operators and avoid sharp jumps in the parameter values.

The use of CGAs in path generation problems needs some justification. For nonredundant manipulators, the number of feasible solutions is finite and depends on the number of joints in the manipulator, the link parameters, and the allowable range of motion of the joints. When the solution is unique, then we have one and only one feasible solution in joint space, which is inherently smooth one; as a result the solution curves representing the joints’ paths through out the evolution process should be smooth. For the multiple solutions case, there exist more than one solution for the problem and switching from one solution corresponding to one robot configuration to another solution corresponding to other robot configuration should not be allowed otherwise the joints’ paths will not be smooth.
and the net displacement of the joints will be large consuming more energy and requiring more time. To illustrate the previous cases, let us consider a 2R planar manipulator, which has two possible solutions for its inverse kinematics problem corresponding to “elbow up” and “elbow down” configurations. Fig. 1(a) and (b) show two possible solutions of the first joint corresponding to some geometric Cartesian path. It is clear that the joint’s paths for the two solutions are inherently smooth. If we assume that the first joint has the limits within the range \([0^\circ, 180^\circ]\), then the first solution is not a feasible one; as a result, the second solution is a unique feasible solution that is inherently smooth as shown in the figure. If we relaxed the joint’s limits for the first joint to be within the range \([-180^\circ, 180^\circ]\), then the two given solutions are feasible, i.e., multiple solutions case. Fig. 1(c) shows a solution, which is mathematically true, but it has multiple switching points between the first and second solutions for the first joint. It is clear that the resulting solution is not smooth and the joint’s displacement is larger than the two other solutions. As a result, while solving such problems, the switching from “elbow up” configuration to the “elbow down” configuration should not be allowed despite the fact that it is still a solution to the problem. The probability of switching between different solutions increases as the number of feasible solutions increases. To avoid the switching phenomenon in such cases, smooth solution curves should be used which guarantee that the resulting
solution corresponds to unique configuration of the manipulator. Regarding redundant manipulators, the number of solutions is infinite and this fact should be utilized to produce solutions with minimum oscillations in the joint paths (smooth solutions) since it results in minimizing the net displacement of the joints. The problems encountered in the previous categories (multiple and infinite solutions) are avoided by using smooth solution curves that operate at the joint path level (global nature) rather than the path point level (local nature) as will be shown later.

In our approach, each individual in the population of \( N_p \) size consists of \( M \) smooth curves where each joint’s path is represented by some smooth curve through the evolution process. Initially, \( N_p \times M \) smooth curves are generated representing the initial population. Pairs of individuals are then crossed with probability \( P_c \). Within that pair of parents, individual joints are crossed over with probability \( P_m \); that is the smooth curve of the \( h \)th joint in the first parent is crossed over with that of the \( h \)th joint in the second parent with \( P_m \) probability. In mutation process, each individual child undergoes mutation with probability \( P_m \). However, individual joints are mutated with probability \( P_m \). The CGAs used in this work consist of the following steps:

1. **Initialization.** An initial population comprising of \( N_p \) smooth individuals is randomly generated in this phase. Two one-dimensional smooth functions are proposed for initializing the population; the first function is a modified Gaussian function:

\[
P_f(h, i) = \theta_{init}(h) + \frac{\theta_{final}(h) - \theta_{init}(h)}{N_k - 1} (i - 1) + A \exp \left( \frac{(i - \mu)^2}{2\sigma^2} \right),
\]

while the second function is a tangent hyperbolic function:

\[
P_f(h, i) = \theta_{init}(h) + (\theta_{final}(h) - \theta_{init}(h)) \times 0.5 \left( 1 + \tanh \left( \frac{h - \mu}{\sigma} \right) \right),
\]

for all \( 1 \leq h \leq M, 2 \leq i \leq N_k - 1 \) in case of fixed end points, or \( 1 \leq i \leq N_k - 1 \) in case of free end points, where \( P_f(h, i) \) is the \( i \)th path point angle of the \( h \)th joint for the \( h \)th parent, \( \theta_{init}(h) \) is the initial angle of the \( h \)th joint at the initial configuration of the end-effector, \( \theta_{final}(h) \) is the final angle of the \( h \)th joint at the final configuration of the end-effector, \( N_k \) is the total number of knots (sampling points) across the Cartesian path, \( A \) represents a random number within the range \([-3R(h), 3R(h)]\), where \( R(h) = |\theta_{init}(h) - \theta_{final}(h)| \), \( \mu \) is a random number within the range \([2, N_k - 1] \) in case of fixed end points, or \([1, N_k - 1] \) in case of free end points, and \( \sigma \) is a random number within the range \([1, 2N_k/6] \).

It is to be noted that in the case of free end points, the initial and final joint angle values are randomly generated within the range of the corresponding joint’s limits, while in the case of fixed end points, they are considered as input parameters of the algorithm.

The upper equations show that the modified Gaussian function is a summation of two different functions; the first function is a ramp function while the second function is a normal Gaussian function. When these two functions are added, they might result in an overshoot or an undershoot, which might be preferred sometimes. The tangent hyperbolic function, on the other hand, does not result in any kind of overshoot or undershoot. For both functions, \( \mu \) specifies the center of the function while \( \sigma \) specifies its degree of dispersion. The two initialization functions are shown in Fig. 2.

2. **Evaluation.** The fitness, a nonnegative measure of quality used as a measure to reflect the degree of goodness of the individual, is calculated for each individual in the population as given in Eq. (6).

3. **Selection.** In the selection process, individuals are chosen from the current population to enter a mating pool devoted to the creation of new individuals for the next generation such that the chance of a given individual to be selected to mate is proportional to its relative fitness. This means that best individuals receive more copies in subsequent generations so that their desirable traits may be passed onto their offspring. This step ensures that the overall quality of the population increases from one generation to the next. Due to the probabilistic nature of selection, the individuals can merely be expected, but not guaranteed, to reproduce in proportion to their fitness.

4. **Crossover.** Crossover provides the means by which valuable information is shared among the
population. It combines the features of two parent individuals, say \( j \) and \( k \), to form two children individuals, say \( L \) and \( L + 1 \), that may have new patterns compared to those of their parents and plays a central role in GAs. The crossover process in our algorithm is expressed as

\[
\begin{align*}
C_L(h, i) &= W(h, i)P_j(h, i) \\
        &+ (1 - W(h, i))P_k(h, i),
\end{align*}
\]

\[
C_{L+1}(h, i) = (1 - W(h, i))P_j(h, i) + W(h, i)P_k(h, i),
\]

\[
W(h, i) = 0.5 \left( 1 + \tanh \left( \frac{i - \mu}{\sigma} \right) \right),
\]

for all \( 1 \leq h \leq M, 2 \leq i \leq N_k - 1 \) in case of fixed end points, or \( 1 \leq i \leq N_k \) in case of free end points, where \( P_j \) and \( P_k \) represent the two parents chosen from the mating pool, \( C_L \) and \( C_{L+1} \) are the two children obtained through crossover process, \( W \) represents the crossover weighting function within the range \([0, 1]\), \( \mu \) is a random number within the range \([2, N_k - 1]\) in case of fixed end points, or \([1, N_k]\) in case of free end points, \( \sigma \) is a random number within the range \([1, N_k/6]\). It is to be noted that \( \sigma \) is chosen such that a complete transition from 0 to 1 or vice versa is achieved at the end points in case of fixed end points.

It is noteworthy that crossover operator in our algorithm is applied with a slight modification to that reported in literature. In our method, pairs of individuals are crossed with probability \( P_c \). Within the pair of parents that should undergo crossover process, individual joints are crossed with probability \( P_{c,i} \); that is the smooth curve of the \( h \)th joint in the first parent is crossed with that of the \( h \)th joint in the second parent with \( P_{c,i} \) probability. If we set \( P_{c,i} \) value to 0.5 and \( P_{c,j} \) to 0.5, then in probability one pair of parents between two pairs is to be crossed, and within that pair, \( M/2 \) of the joints will be crossed. If we set \( P_{c,j} \) value to unity, then each joint in the first parent is crossed with the similar joint (first with first, second with second, and so on) in the second parent. Fig. 3 shows the crossover process between two random joints’ paths corresponding to the \( h \)th joint in the first parent and the \( h \)th joint in the second parent. It is clear that new information is incorporated in the children while maintaining the smoothness of the resulting paths.

5. Mutation: Mutation is often introduced to guard against premature convergence. Generally, over a period of several generations, the gene pool tends to become more and more homogeneous. The purpose
of mutation is to introduce occasional perturbations to the parameters to maintain genetic diversity within the population. The mutation process in our algorithm is governed by the following formulas:

\[ C_m^j(h,i) = C_j(h,i) + dM(h,i), \]  

(12)

where \( C_j \) represents the \( j \)th child produced through the crossover process, \( C_m^j \) is the mutated \( j \)th child, \( M \) is the Gaussian mutation function, \( d \) represents

\[ M(h,i) = \exp\left( -\frac{(i - \mu)^2}{2\sigma^2} \right), \]  

(13)
a random number within the range $[-\text{range}(h), \text{range}(h)]$, $\text{range}(h)$ represents the difference between the minimum and maximum values of the child $C_j$, and $\mu$ and $\sigma$ are as given in the crossover process.

In mutation process, each individual child undergoes mutation with probability $P_{mu}$. However, for each child that should undergo mutation process, individual joints are mutated with probability $P_{mj}$. If we set $P_{mu}$ value to 0.5 and $P_{mj}$ to 0.5, then in probability one child between two children is to be mutated, and within that child, $M/2$ of the joints’ paths will be mutated. Fig. 4 shows the mutation process in a joint’s path of certain child. As in crossover process, some new information is incorporated in the children while maintaining the smoothness of the resulting paths.

6. Replacement. After generating the offspring’s population through the application of the genetic operators to the parents’ population, the parents’ population is totally replaced by the offspring’s population. This is known as nonoverlapping, generational, replacement. This completes the “life cycle” of the population.

7. Termination. The GA is terminated when some convergence criterion is met. Possible convergence criteria are: the fitness of the best individual so far found exceeds a threshold value, the maximum number of generations is reached, or the progress limit, the improvement in the fitness value of the best member of the population over a specified number of generations is less than some predefined threshold, is reached. After terminating the algorithm, the optimal
solution of the problem is the best individual so far found.

The determination of a set of feasible joints’ paths is required rather than the determination of a set of possible joints’ paths since any possible solution of the inverse kinematics problem might not be feasible due to the limits on the angles that can be reached physically by each joint. For example, PUMA 560 can reach certain goals with eight different possible solutions, but due to the limits on the joints’ angles, some of these eight solutions might not be physically feasible or accessible. As a result, the extrema of all joints’ paths must be within the geometric limits of each joint. In order to achieve this goal, a regular check should be done on the joint’s values of every joint such that they will not exceed the given joints’ limits of the manipulator during the evolution process. To be more specific, the initialization and mutation processes should have this regular check on joints’ values. Regarding the initialization process, this check for parents is required if we are using the modified Gaussian function due to any possible overshoot or undershoot that might go beyond the limits, while in the case of tangent hyperbolic function, this step is not required. For the mutation process, the check for joints’ values of the children is required due to the fact that the joints’ angles might exceed the joints’ limits after the addition/subtraction of the mutation function. Regarding the crossover process, if the regular check is performed in the initialization and mutation processes then this step is not required in the crossover process due to the nature of the crossover function.

For this purpose, we have proposed two methods that maintain the smoothness of the joints’ paths; the fixed limit method and the mirror limit method. The mirror limit method is governed by the following equation:

\[
P_j(h,i) = \begin{cases} 
P_j(h,i), & \theta_{\text{lower}}(h) \leq P_j(h,i) \leq \theta_{\text{upper}}(h), \\
\theta_{\text{upper}}(h) - (P_j(h,i) - \theta_{\text{upper}}(h)), & P_j(h,i) > \theta_{\text{upper}}(h), \\
\theta_{\text{lower}}(h) + (\theta_{\text{upper}}(h) - P_j(h,i)), & P_j(h,i) < \theta_{\text{lower}}(h). 
\end{cases}
\]

The fixed limit method, on the other hand, is governed by the following equation:

\[
P_j(h,i) = \begin{cases} 
P_j(h,i), & \theta_{\text{lower}}(h) \leq P_j(h,i) \leq \theta_{\text{upper}}(h), \\
\theta_{\text{upper}}(h), & P_j(h,i) > \theta_{\text{upper}}(h), \\
\theta_{\text{lower}}(h), & P_j(h,i) < \theta_{\text{lower}}(h). 
\end{cases}
\]

The two methods are illustrated in Fig. 5 where it is assumed that the upper limit of certain joint equals 180°. As stated previously, the two methods result in smooth joint’s path.

To summarize the evolution process, an individual is a candidate solution of the joints’ angles; i.e. each individual consists of \( N_j \) joints’ paths each consisting of \( N_k \) path points, this results in a two-dimensional array of the size \( M \times N_k \). The population undergoes the selection process, which results in a mating pool among which pairs of individuals are crossed with probability \( P_c \). Within that pair of parents, individual joints are crossed over with probability \( P_m \). This process results in an offspring’s generation where every individual child undergoes mutation with probability \( P_m \). Within that child, individual joints are mutated with probability \( P_m \). After that, the next generation is produced according to the replacement strategy applied.

This process is repeated till the convergence criterion is met where the \( M \times N_k \) parameters of the best individual are the required joints’ angles. The final goal of discovering the required joints’ paths is translated into finding the fittest individual in genetic terms. The block diagram of the CGA is given in Fig. 6.

Two additional operators were introduced to enhance the performance of the GA, the elitism operator and the “extinction and immigration” operator. The preservation of the best solution or solutions, elitism, and moving it to the next generation is vital to the effectiveness of the GA [5]. It is utilized to ensure that the fitness of the best candidate solution in the current population must be larger than or equal to that of the previous population (i.e., a good solution found should not be lost through some of the genetic operators). In other words, it guarantees that the best-fitness in the population is a monotonically nonincreasing function. Extinction and immigration operator [10] is applied when all individuals in the population are identical or when the improvement in the fitness value of the best individual over a certain number of generations is less than some threshold value. The number of individuals
in the population associated with better fitness grows exponentially [7]. Therefore, after some generations, the mating pool will consist of almost identical members. This means that no new information will be obtained. The GA thus tends to stagnate; "extinction and immigration" operator is used to bypass this difficulty. This operator, as indicated by its name, consists of two stages; the first stage is the extinction process where all of the individuals in the current generation are removed except the best-of-generation individual. The second stage is the mass-immigration process where the extinct population is filled out again by generating \( N_p - 1 \) individuals to keep the population size fixed. The generated population is divided into two equal segments each of \( N_p/2 \) size; the first segment, with \( 2 \leq j \leq N_p/2 \), is generated as in the initialization phase, while the other segment is generated by performing continuous mutations to the best-of-generation individual as given by the formula:

\[
P_j(h, i) = P_1(h, i) + dM(h, i),
\]

\[
\frac{1}{2}N_p + 1 \leq j \leq N_p.
\]  

(16)

where \( P_j \) is the \( j \)th parent generated using immigration operator, \( P_1 \) represents the best-of-generation individual, \( M \) is the Gaussian mutation function, and \( d \) represents a random number within the range \([-\text{range}(h), \text{range}(h)]\). \( \text{range}(h) \) represents the difference between the minimum and maximum values of the best individual.

Fig. 5. Methods used to limit joints’ angles.

Fig. 6. Block diagram of the CGA.
4. Simulation results

The proposed technique is used to solve the Cartesian path generation problem for several manipulators. The input data to the algorithm is divided into two parts; the genetic-algorithm related parameters and the problem related parameters. The genetic-algorithm related parameters include the population size, \( N_p \), the initialization function, the individual crossover probability, \( P_{cx} \), the joint crossover probability, \( P_{cj} \), the individual mutation probability, \( P_{mut} \), the joint mutation probability, \( P_{ja} \), the method used when the joints’ angles exceed the joint limit, the immigration threshold value, the corresponding number of generations, and finally the termination criterion. The problem related parameters include the link parameters, the number of joints in the manipulator, \( M \), the robot’s degrees of freedom, \( N \), the number of path points, \( N_p \), the joints’ limits \( \theta_{lim} \) and \( \theta_{max} \) for \( h = 1, \ldots, M \), and the desired Cartesian path \( (P_{k}(i), \ldots, P_{k}(N)) \). Regarding the initial and final joints’ angles \( \theta_{initial} \) and \( \theta_{final} \) for \( h = 1, \ldots, M \), there are two cases; in the fixed end points case, these values are fed to the algorithm as input parameters using either closed-form inverse kinematics formulas or any numerical technique, while in the free end points case, the end points are not considered as input parameters to the algorithm since they are not given. The initial settings of the GA parameters are as follows: the population size is set to 500 individuals. The tangent hyperbolic function given in Eq. (8) is used in the initialization phase. The rank-based selection strategy is used where the rank-based ratio is set to 0.5. The individual crossover probability is kept at 0.5, the joint crossover probability is also set to 0.5. The individual mutation probability and the joint mutation probability are kept at 0.5. The fixed limit method given in Eq. (15) is used to limit the joints’ angles. Generational replacement scheme is applied where the number of elite parents that are passed to the next generation is one-tenth of the population. Extinction and immigration operator is applied when the improvement in the fitness value of the best individual over 400 generations is less than 0.01. The genetic algorithm is stopped when one of the following conditions is met. First, the fitness of the best individual of the population reaches a value of 0.99, i.e. the accumulative deviation of the end-effector, \( E \), of the best individual is less than or equal to 0.01. Second, the maximum deviation at any path point of the best individual is less than or equal to 0.001. Third, a maximum number of 10,000 generations is reached. Fourth, the improvement in the fitness value of the best individual in the population over 1000 generations is less than 0.01. It is to be noted that the first two conditions indicate to a successful termination process (optimal solution is found), while the last two conditions point to a partially successful end depending on the fitness of the best individual in the population (near-optimal solution is reached).

Throughout this paper, the convergence speed of the algorithm, whenever used, means the average number of generations required for convergence. All the results used were obtained using P-I 350 MHz computer and the CGA is written in Visual Basic environment. Due to the stochastic nature of GAs, 12 different runs were made for every result obtained in this work using a different random number generator seed; results are the average values whenever possible.

The path generation problems considered in this work may be divided into three main categories depending on the number of solutions obtained:

1. **Unique solution.** This is the “fixed end points” case of nonredundant manipulators where the initial and final joints’ angles corresponding to the initial and final configurations of the end-effector are given using the existing closed-form solution for the inverse kinematics of the manipulator or using any numerical method. Out of the existing inverse kinematics solutions for the end points, the one corresponding to the minimum displacement of all joints is fed to the algorithm and given these joints’ angles, a unique smooth solution curve connecting them is available.

2. **Multiple solutions.** This is the “free end points” case of nonredundant manipulators where the initial and final joints’ angles corresponding to the initial and final configurations of the end-effector are not given (i.e., the closed-form solution for the inverse kinematics of the manipulator or any numerical method is not used at all). The algorithm automatically finds one smooth solution out of the existing solutions every time it is executed. The number of multiple solutions depends on the number of joints in the manipulator, the link parameters,
and the allowable range of motion of the joints. In general, the more nonzero link parameters there are, the more ways there will be to reach a certain goal and the larger the number of solutions. For a completely general rotary-jointed manipulator with six degrees of freedom, there are up to 16 solutions possible [1].

3. Infinite number of solutions. This is the case of redundant manipulators where redundancy results in infinite number of solutions. The initial and final joints’ angles corresponding to the initial and final configurations of the end-effector may be given (fixed end points) or not (free end points), but they are generally not given. The algorithm automatically finds one smooth solution out of the infinite number of solutions every time it is executed.

Each of the above mentioned categories would be covered in details in this paper. For the unique and multiple solutions categories, we have selected 3R planar manipulator. Finally, we will apply the algorithm for the solution of the path generation problem for a general 6R serial manipulator as given in the mutation process. For this case, $N = 3$, $M = 3$ and the forward kinematics model is given in [1].

The desired geometric path is of sinusoidal shape as given by

\[ s_{\text{final}} = 0.1, \quad s_{\text{initial}} = 0.1 + \frac{1}{4}\pi. \]

\[ P_0(1, i) = X_0(i) = s_{\text{initial}} + \frac{s_{\text{final}} - s_{\text{initial}}}{N_k - 1}(i - 1), \]

\[ P_0(2, i) = Y_0(i) = L_1 \sin(\theta_1(i)) + L_2 \sin(\theta_1(i) + \theta_2), \]

\[ P_0(3, i) = Z_0(i) = 0, \quad 1 \leq i \leq N_k. \]
Table 1

<table>
<thead>
<tr>
<th>Joint number (h)</th>
<th>(a_{h-1}) ((°))</th>
<th>(a_{h-1})</th>
<th>(d_{h})</th>
<th>(\theta_{h})</th>
<th>(\theta_{\text{min}}(h)) ((°))</th>
<th>(\theta_{\text{max}}(h)) ((°))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(\theta_1)</td>
<td>–160</td>
<td>160</td>
</tr>
<tr>
<td>2</td>
<td>–90</td>
<td>0.4318</td>
<td>0.14909</td>
<td>(\theta_2)</td>
<td>–225</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.02032</td>
<td>0.43307</td>
<td>(\theta_3)</td>
<td>–45</td>
<td>225</td>
</tr>
<tr>
<td>4</td>
<td>–90</td>
<td>0</td>
<td>0</td>
<td>(\theta_4)</td>
<td>–110</td>
<td>170</td>
</tr>
<tr>
<td>5</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>(\theta_5)</td>
<td>–100</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>–90</td>
<td>0</td>
<td>0</td>
<td>(\theta_6)</td>
<td>–266</td>
<td>–266</td>
</tr>
</tbody>
</table>

Fig. 7. Evolutionary progress plot for the best-of-generation individual regarding the unique solution category using the random mutation center and the random dispersion factor methods for (a) PUMA manipulator, and (b) 2R planar manipulator.
The initial configuration of the end-effector along the Cartesian path is given as \((0.1, 0.25, 0)\), while the final configuration of the end-effector along the Cartesian path is given as \((0.1 + \pi/8, 0.25)\). For the PUMA manipulator, the \(z\)-coordinates are kept at zero values while this degree of freedom is not taken into consideration for the planar case. The number of path points, \(N_k\), is 100 points. The initial and final joints’ angles corresponding to the initial and final configurations of the end-effector along the Cartesian path are found using the inverse kinematics model of the corresponding manipulator. This process will result in a unique smooth solution for the path generation problem.

The deviation between the desired path and the generated path at some general path point, \(i\), is given as

\[
E(i) = \sum_{k=1}^{N_k} |P_{dc}(k,i) - P_{gc}(k,i)|.
\]

The accumulative deviation between the desired path and the generated path is given as

\[
E = \sum_{i=1}^{N_s - 1} \sum_{k=1}^{N_k} |P_{dc}(k,i) - P_{gc}(k,i)| = \sum_{i=2}^{N_s} E(i).
\]

The evolutionary progress plots, EPPs, of the best-fitess individual for the PUMA and 2R manipulators using the random mutation center and the random dispersion factor methods are shown respectively in Fig. 7. The best-of-generation fitness value has an upward trend but levels off quickly. The algorithm takes about 10–15% of the total number of generations to reach a near-optimal solution with 0.8 fitness value, while it takes about 85% of the total number of generations to reach the optimal solution from near-optimal one. This is sometimes called the “hare strategy” as opposed to the “tortoise strategy” of slow and steady increase. This shows how the GA finds the near-optimal solutions very early in the GA run. It is also clear that the PUMA manipulator takes about 1.5 the number of generations of the 2R manipulator to reach the optimal solution.

The path point deviations for both manipulators are given in Fig. 8, which shows that the average path point deviation is almost 0.0002 for the PUMA manipulator and 0.00025 for the 2R manipulator. The desired and generated Cartesian paths are given in Fig. 9 for the PUMA manipulator while they are shown in Fig. 10 for the 2R manipulator. It is clear from the previous figures that the desired and the generated Cartesian paths are almost the same with extremely low deviation level. This gives an indication about how accurate the proposed algorithm is in generating the paths. The joints’ paths for the first, second and third joints of the PUMA manipulator are shown in Fig. 11 while the joints’ paths for the first and second joints of the 2R manipulator are shown in Fig. 12. It is obvious that the resulting solution curves in joint space are inherently smooth. In order to show the accuracy of the CGA in generating the paths in joint space, we have solved the same problem using the already existing closed-form solutions for both manipulators. The difference between the analytical solutions and the solutions provided by CGA for the three joints of the PUMA manipulator are given in Fig. 13. It is clear that the average difference between the two solutions is about 0.025. The difference between the analytical solutions and the solutions provided by CGA for the two joints of the 2R manipulator are given in Fig. 14 which shows that the average difference between the two solutions is about 0.025. This high accuracy gives us confidence about the behavior of the algorithm for the solution of path generation problems of manipulators that might not have a derived closed-form solution.

After that, an analysis on the various methods used for generating the mutation center and the dispersion factor previously discussed is performed. The EPPs for the PUMA manipulator using the random and knowledgeable dispersion factors respectively are given in Fig. 15. It is clear that for a certain dispersion factor method, the use of the deterministic mutation center greatly speeds up the convergence process; the number of generations required for convergence has dropped to almost one-fifth of that of the random mutation center. The Lamarckian method is generally between the random and the deterministic methods. Furthermore, the knowledgeable dispersion factor method has faster convergence speed than that of the random dispersion factor method for a certain mutation center scheme. As a result, the best methods are the deterministic mutation center and the knowledgeable dispersion factor, which were able to reach an optimal solution within 400 generations in 210 s as compared to 2500 generations for the random methods. The EPPs for the 2R manipulator using the random and knowledgeable dispersion factors respectively are given in
Fig. 16 where similar observations are concluded. This makes the deterministic mutation center and the knowledgeable dispersion factor as the algorithm’s default methods for the unique solution category.

It is noteworthy that more accurate results can be obtained by changing the convergence criteria of the algorithm depending on the tracking capabilities of the manipulator. For example, when the second termination condition is only used as the convergence criterion and the maximum deviation at any path point of the best individual is adjusted to be less than or equal to 0.0001 (i.e., 0.1 mm assuming that the manipulator has a tracking capability of this value), the algorithm converges to an average fitness value of 0.9981 and the average path point deviation obtained is 18 μm for the PUMA manipulator. However, the cost to be paid in this case is the number of generations required to reach this optimal solution, since it took 1200 generations. This means that to go from an accuracy of 0.2 mm to 0.018 mm, the required number of generations for convergence increases from about 400 to 1200 generations. Regarding the 2R manipulator, the algorithm converges to an average fitness value of 0.9980 within 650 generations and the average path
point deviation obtained is 19 μm. This goes in agreement with the previous conclusion that the algorithm takes large number of generations in the fine-tuning stage.

After that, the way in which the joints’ angles evolve for all the joints of PUMA and 2R manipulators is studied. Fig. 17 shows the evolution of three different path points corresponding to left most, center and right most path points \((i = 2, i = 50 \text{ and } i = 99)\) for the three joints of the PUMA manipulator while Fig. 18 shows the evolution of the two joints of the 2R manipulator for the same path points. It is observed that the near-optimal solution is reached very early during GA evolution (between 50 and 100 generations).

The effect of the two initialization functions, previously discussed, on the convergence speed of the algorithm is covered next. For this purpose, we have introduced an additional Cartesian path generation problem, which is of straight line shape as given by

\[
X_{\text{initial}} = 0.0, \quad X_{\text{final}} = 0.25, \\
Y_{\text{initial}} = 0.0, \quad Y_{\text{final}} = 0.25, \\
Z_{\text{initial}} = 0.0, \quad Z_{\text{final}} = 0.25, \quad 1 \leq i \leq N_k.
\]  

Table 2 shows that the initialization functions have minor effect on the convergence speed of the algorithm because usually the effect of the initial population dies after few tens of generations and the convergence speed after that is governed by the selection mechanism, crossover and mutation operators. However,
Fig. 10. Desired and generated Cartesian path for 2R manipulator in (a) X-plane, and (b) Y-plane.

Table 2
The effect of the initialization function used on the convergence speed of the algorithm for the unique solution category

<table>
<thead>
<tr>
<th>Manipulator name</th>
<th>Problem number</th>
<th>Initialization function</th>
<th>Average number of generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUMA</td>
<td>First</td>
<td>Tangent hyperbolic</td>
<td>302</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified Gaussian</td>
<td>373</td>
</tr>
<tr>
<td></td>
<td>Second</td>
<td>Tangent hyperbolic</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified Gaussian</td>
<td>141</td>
</tr>
<tr>
<td>2R</td>
<td>First</td>
<td>Tangent hyperbolic</td>
<td>138</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified Gaussian</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td>Second</td>
<td>Tangent hyperbolic</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified Gaussian</td>
<td>78</td>
</tr>
</tbody>
</table>
the tangent hyperbolic function generally results in faster convergence speed than the modified Gaussian function for the two path generation problems. As a result, the tangent hyperbolic function will be used as the algorithm’s default function for the unique solution category.

The influence of the two methods used in our algorithm to limit the joints’ angle values on the convergence speed of the algorithm is also studied. Table 3 shows that both of the fixed limit method and the mirror limit method converge in almost the same number of generations. For the first problem, the fixed limit method has faster convergence speed than the mirror limit method while the situation is reversed for the second problem. As a result, any one of them might be chosen as the algorithm’s default method.

![Fig. 11. Joints’ paths of PUMA manipulator for (a) first joint, (b) second joint, and (c) third joint.](image)

<table>
<thead>
<tr>
<th>Manipulator</th>
<th>Problem number</th>
<th>Limit method</th>
<th>Average number of generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUMA</td>
<td>First</td>
<td>Mirror</td>
<td>286</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fixed</td>
<td>275</td>
</tr>
<tr>
<td></td>
<td>Second</td>
<td>Mirror</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fixed</td>
<td>147</td>
</tr>
<tr>
<td>2R</td>
<td>First</td>
<td>Mirror</td>
<td>145</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fixed</td>
<td>137</td>
</tr>
<tr>
<td></td>
<td>Second</td>
<td>Mirror</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fixed</td>
<td>97</td>
</tr>
</tbody>
</table>

Table 3: The effect of the limit method used on the convergence speed of the algorithm for the unique solution category.
4.2. Multiple solutions category

For this category, we will use the same manipulators and the same path generation problems previously discussed in the previous category. The only difference between this category and the previous one is that in this category the initial and final joints’ angles corresponding to the initial and final configurations of the end-effector along the Cartesian path are not given at all (free end points) while in the previous one, the initial and final joints’ angles are given in advance (fixed end points) using either the closed-form solution or by any numerical technique. This process will result in one or more solutions for the path generation problem depending on the robot configuration. The deviation between the desired path and the generated path at some general path point, $i$, is given as

$$E(i) = \sum_{k=1}^{N} |P_{dc}(k, i) - P_{gc}(k, i)|.$$  \hspace{1cm} (22)
The accumulative deviation between the two paths is given as
\[ E = \sum_{i=1}^{N_k} \sum_{k=1}^{N} |P_{dc}(k, i) - P_{gc}(k, i)| = \sum_{i=1}^{N_k} E(i). \] (23)

From a computational point of view, it is observed that the current category finds the path point deviations at two additional path points \((i = 1, N_k)\) as compared to the previous category where these two points are not included since the corresponding joints’ angles have zero path point deviation.

The evolutionary progress plots of the best-fitness individual for the PUMA and the 2R manipulators using random mutation center and random dispersion factor methods are shown respectively in Fig. 19. It is noted that the algorithm takes about 400 generations for the PUMA manipulator and about 150 generations for the 2R manipulator to reach a near-optimal solution with 0.8 fitness value, while it takes about 2800 generations for the PUMA manipulator and about 1700 generations for the 2R manipulator to reach the optimal solution from near-optimal one. This shows that most of the computational burden of the algorithm is spent in the fine-tuning stage while the near-optimal solutions are found very early in the GA run. It is also clear that the PUMA manipulator takes about 1.5 the number of generations of the 2R manipulator to reach the optimal solution. As compared to the unique solution category, the PUMA manipulator previously re-
quired almost 2500 generations for convergence while it requires in this category about 3300 generations for convergence. The 2R manipulator previously required almost 1600 generations for convergence while it requires in this category about 1900 generations for convergence. This shows that the multiple solutions category is a tougher problem for the algorithm than the unique solution category since the number of solutions is still finite while the end points are not given in the multiple solutions category. The path point deviations for both manipulators are given in Fig. 20, which shows that the average path point deviation is almost 0.0002 for the PUMA manipulator and 0.00015 for the 2R manipulator. This gives an indication about how accurate the proposed algorithm is in generating the paths. The desired and generated Cartesian paths are as given previously in Figs. 9 and 10 for PUMA and 2R manipulators, respectively.

For PUMA 560 manipulator, it is known that it can reach certain goals with four different solutions when position is only taken into account, but due to the limits on the joints’ angles, some of these four solutions
may not be feasible [1]. The 2R manipulator, on the other hand, has two possible solutions. In our case, two feasible solutions are obtained for the PUMA manipulator since two other solutions out of the four violate the joints’ limits constraints, while two feasible solutions are obtained for the 2R manipulator. The joints’ paths for the first solution of the PUMA manipulator are shown in Fig. 21 while the joints’ paths for the second solution of the PUMA manipulator are shown in Fig. 22. The first solution obtained is the same one obtained previously in the unique solution category as given in Fig. 11. The joints’ paths for the first solution of the 2R manipulator are shown in Fig. 23 while the joints’ paths for the second solution of the 2R manipulator are shown in Fig. 24. The first solution obtained is the same one obtained previously in the unique solution category as given in Fig. 12. It is obvious that the resulting curves in joint space are smooth and do not have any switching between the multiple solutions.
The frequency at which the solutions are obtained should not be neglected. Out of the 12 runs executed for each result obtained in this paper, 10 runs result in the first solution while two runs result in the second solution for PUMA manipulator. The ranges of the joints' angles for the first solution are given as following: \([-47^\circ, 12^\circ]\) for the first joint, \([-75^\circ, -48^\circ]\) for the second joint and \([63^\circ, 10^\circ]\) for the third joint. The ranges of the joints' angles for the second solution are given as following: \([-78^\circ, -137^\circ]\) for the first joint, \([-104^\circ, -132^\circ]\) for the second joint and \([123^\circ, 176^\circ]\) for the third joint. The joints' limits are \([-160^\circ, 160^\circ]\) for the first joint, \([-225^\circ, 45^\circ]\) for the second joint and \([-45^\circ, 225^\circ]\) for the third joint. It is clear that the second solution is closer to the joints' limits than the first solution. This means that the algorithm tries to find safe solutions (away from joints' limits) more frequent than other solutions, which are close to the joints' limits. For the 2R manipulator, 11 runs result in the first solution while single
run results in the second solution. The ranges of the joints’ angles for the first solution are \([5^\circ, -29^\circ]\) for the first joint and \([148^\circ, 108^\circ]\) for the second joint. The ranges of the joints’ angles for the second solution are \([142^\circ, 83^\circ]\) for the first joint and \([-148^\circ, -108^\circ]\) for the second joint. The joints’ limits are \([-180^\circ, 180^\circ]\) for both joints. It is clear that the second solution is closer to the joints’ limits than the first solution. This goes in agreement with the previous conclusion, which states that the algorithm tries to find the safe solutions (away from joints’ limits) more frequent than solutions that are close to the joints’ limits.

An analysis on the various methods used for generating the mutation center and the dispersion factor is performed next. The EPPs for the PUMA manipulator using the random and knowledgeable dispersion factors respectively are given in Fig. 25. It is clear that for a certain dispersion factor method, the use of the deterministic mutation center speeds up the convergence process; the number of generations required for convergence has dropped to almost one-fifth of that of the random mutation center. The Lamarckian method comes in the second rank after the deterministic method while the random method is the slowest one. It is also noted that the use of the knowledgeable dispersion factor method has faster convergence speed than that of the random dispersion factor method for a certain mutation center method. As a result, the best methods are the deterministic mutation center and the knowledgeable dispersion factor, which were able to reach an optimal solu-
tion within 500 generations in 320 s as compared to 3200 generations for the random methods. The EPPs for the 2R manipulator using the random and knowledgeable dispersion factors respectively are given in Fig. 26 where similar observations are concluded. This makes the deterministic mutation center and the knowledgeable dispersion factor as the algorithm’s default methods for the multiple solutions category.

As compared to the unique solution category, Figs. 25 and 26 show that for all variations in mutation center and dispersion factor, the current category requires larger number of generations for convergence than the previous one. It is noteworthy that more accurate results can be obtained by changing the convergence criteria of the algorithm, i.e., when the second termination condition is only used as the convergence criterion and the maximum deviation at any path point of the best individual is adjusted to be less than or equal to 0.0001, the algorithm converges to an average fitness value of 0.9979 within 1400 generations and the average path point deviation obtained is 20 μm in case of PUMA manipulator. For the 2R manipulator, the algorithm converges to an
average fitness value of 0.9981 within 600 generations and the average path point deviation obtained is 19 μm.

After that, the way in which the joints’ angles evolve for PUMA and 2R manipulators is studied. Fig. 27 shows the evolution of three different path points \((i = 1, i = 50 \text{ and } i = 100)\) for the three joints of the PUMA manipulator while Fig. 28 shows the evolution of the same path points for the two joints of the 2R manipulator. It is observed that the near-optimal solution is reached within 100 generations while the remaining number of generations is spent in the fine-tuning stage. In addition to that, the initial generations show some random oscillations. As compared to the unique solution category, the magnitude of the oscillations in the current category is larger than that of the previous category. This fact is expected since the end points in the current category are free and the algorithm should explore the search space within the joints’ limits in the initial stages of the GA run in order to reach a near-optimal solution. This is the reason behind the fact that the
The multiple solutions category is considered as a tougher problem for the algorithm than the unique solution category. The effect of the two initialization functions given in Eqs. (7) and (8) on the convergence speed of the algorithm is covered next with the same path generation problems previously introduced in the unique solution category. Table 4 shows that the initialization functions have minor effect on the convergence speed of the algorithm. However, the modified Gaussian function results in faster convergence speed than that of the tangent hyperbolic function for the first problem while the tangent hyperbolic function results in faster convergence speed than that of the modified Gaussian function for the second problem. As a result, any of the two functions may be used as the algorithm’s default function for the multiple solutions category.

The influence of the two methods used in our algorithm to limit the joints’ angle values is also studied. Table 5 shows that both of the fixed limit method and the mirror limit method converge in al-
Table 4
The effect of the initialization function used on the convergence speed of the algorithm for the multiple solutions category

<table>
<thead>
<tr>
<th>Manipulator name</th>
<th>Problem number</th>
<th>Initialization function</th>
<th>Average number of generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUMA</td>
<td>First</td>
<td>Tangent hyperbolic</td>
<td>347</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified Gaussian</td>
<td>330</td>
</tr>
<tr>
<td></td>
<td>Second</td>
<td>Tangent hyperbolic</td>
<td>158</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified Gaussian</td>
<td>234</td>
</tr>
<tr>
<td>2R</td>
<td>First</td>
<td>Tangent hyperbolic</td>
<td>149</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified Gaussian</td>
<td>146</td>
</tr>
<tr>
<td></td>
<td>Second</td>
<td>Tangent hyperbolic</td>
<td>114</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified Gaussian</td>
<td>132</td>
</tr>
</tbody>
</table>

Fig. 21. Joints’ paths of the first solution of PUMA manipulator for (a) first joint, (b) second joint, and (c) third joint.
Fig. 22. Joints’ paths of the second solution of PUMA manipulator for (a) first joint, (b) second joint, and (c) third joint.

Table 5
The effect of the limit method used on the convergence speed of the algorithm for the multiple solutions category

<table>
<thead>
<tr>
<th>Manipulator name</th>
<th>Problem number</th>
<th>Limit method</th>
<th>Average number of generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUMA</td>
<td>First</td>
<td>Mirror</td>
<td>334</td>
</tr>
<tr>
<td></td>
<td>Fixed</td>
<td>302</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Second</td>
<td>Mirror</td>
<td>182</td>
</tr>
<tr>
<td></td>
<td>Fixed</td>
<td>172</td>
<td></td>
</tr>
<tr>
<td>2R</td>
<td>First</td>
<td>Mirror</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>Fixed</td>
<td>155</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Second</td>
<td>Mirror</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>Fixed</td>
<td>112</td>
<td></td>
</tr>
</tbody>
</table>
most the same number of generations. However, the fixed limit method has generally faster convergence speed than that of the mirror limit method. As a result, the fixed limit method might be chosen as the algorithm’s default method for the multiple solutions category.

4.3. Infinite solutions category

For this category, we will use 3R planar redundant manipulator with $L_1 = L_2 = L_3 = 0.5$ m. For this case, $N = 2, M = 3, \theta_{\text{lower}}(h) = -180^\circ$ and $\theta_{\text{upper}}(h) = 180^\circ$ for $h = 1, 2, 3$. The forward kinematics model of the manipulator is used to find the generated Cartesian path as given by

$$X_{gc}(i) = \sum_{j=1}^{M} L_j \cos \left( \sum_{k=1}^{j} \theta(k, i) \right),$$

$$Y_{gc}(i) = \sum_{j=1}^{M} L_j \sin \left( \sum_{k=1}^{j} \theta(k, i) \right),$$

where $1 \leq i \leq N_k$.
The same path generation problems previously discussed will be used in this category. The initial and final joints’ angles corresponding to the initial and final configurations of the end-effector along the Cartesian path are not given (free end points). The path point deviation and the accumulative deviation between the desired and generated paths are as given in Eqs. (22) and (23) where the path point deviations at the end points are included.

The evolutionary progress plot of the best-fitness individual using the random mutation center and random dispersion factor methods and the corresponding path point deviation are given in Fig. 29. It is noted that the algorithm converges within about 500 generations and the average path point deviation is almost 0.0002. As compared to the evolutionary progress plots of the previous two categories, it is obvious that this category is the simplest category for the algorithm since the number of solutions is infinite in this category while it is either unique or finite in the previous categories.

The joints’ paths for one solution out of the 12 solutions obtained in the 12 runs are shown in Fig. 30. It is obvious that the resulting curves in joint space are smooth. In addition to that, it is clear that the solution...
is away from joints’ limits that are $[-180^\circ, 180^\circ]$ for all joints. This goes in agreement with the previous conclusion, which states that the algorithm tries to find safe solutions.

After that, an analysis on the various methods used for generating the mutation center and the dispersion factor previously discussed is performed. The EPPs given in Fig. 31 show that for a certain dispersion factor method, the deterministic mutation center has the fastest convergence speed. The random mutation center has the slowest convergence speed while the Lamarckian method is midway between the random and the deterministic methods. Furthermore, the knowledgeable dispersion factor method has faster convergence speed than that of the random dispersion factor method for a certain mutation center method.

As a result, the best methods are the deterministic mutation center and the knowledgeable dispersion factor, which were able to reach an optimal solution within almost 150 generations in 90 s. This makes the
deterministic mutation center and the knowledgeable dispersion factor as the algorithm’s default methods for the infinite solutions category. Furthermore, the EPPs show that the current category is the fastest in convergence speed as compared to the previous two categories, i.e., this category is the simplest to the algorithm. When the second termination condition was only used as the convergence criterion of the algorithm and the maximum deviation at any path point of the best individual was adjusted to be less than or equal to 0.0001, the algorithm converged to an average fitness value of 0.9945 and the average path point deviation obtained was 55 μm. However, the cost to be paid in this case is the number of generations required to reach this optimal solution, since it took 500 generations. This means that to go from an accuracy of 0.2 mm to 0.055 mm, the required number of generations for convergence increases from about 150 to 500 generations.
After that, the way in which the joints’ angles evolve for all the joints is studied. Fig. 32 shows the evolution of three path points \( (i = 1, i = 50 \text{ and } i = 100) \) for the three joints of the manipulator. It is observed that the Coarse-tuning stage lasts for almost 80 generations while the remaining number of generations is spent in the fine-tuning stage. In addition to that, the initial generations show some random oscillations. As compared to the unique and multiple solutions categories, the magnitude of the oscillations in the current category is the largest since the end points in the current category are free and the number of solutions is infinite.

The effect of the two initialization functions, previously discussed, on the convergence speed of the algorithm is covered next with the same path generation problems previously introduced in the unique solution category. Table 6 shows that the tangent hyperbolic function results in faster convergence speed than that of the modified Gaussian function for both problems. As a result, the tangent hyperbolic function is used as the algorithm’s default function for this category.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Initialization function</th>
<th>Average number of generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>Tangent hyperbolic</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>Modified Gaussian</td>
<td>250</td>
</tr>
<tr>
<td>Second</td>
<td>Tangent hyperbolic</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Modified Gaussian</td>
<td>121</td>
</tr>
</tbody>
</table>
Fig. 28. Evolution of different path points of 2R manipulator for (a) first joint, and (b) second joint.

The influence of the two methods used in our algorithm to limit the joints’ angle values is also studied. Table 7 shows that both of the fixed limit method and the mirror limit method converge in almost the same number of generations. As a result, any of the two methods may be used as the algorithm’s default method for the infinite solutions category.

### 4.4. General 6R manipulator

The last step in our work is to explore the ability of the proposed algorithm in solving the path generation

<table>
<thead>
<tr>
<th>Problem number</th>
<th>Limit method</th>
<th>Average number of generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>Mirror</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>Fixed</td>
<td>115</td>
</tr>
<tr>
<td>Second</td>
<td>Mirror</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Fixed</td>
<td>80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem number</th>
<th>Limit method</th>
<th>Average number of generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>Mirror</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>Fixed</td>
<td>115</td>
</tr>
<tr>
<td>Second</td>
<td>Mirror</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Fixed</td>
<td>80</td>
</tr>
</tbody>
</table>
problems of some general 6R manipulator. The link parameters and the joints’ limits of the manipulator are given in Table 8. The homogeneous transform that defines frame \( \{ h \} \) relative to \( \{ h-1 \} \) is given as

\[
\begin{bmatrix}
    c\theta_{h} - s\theta_{h} & 0 & -s\alpha_{h} - c\alpha_{h} - s\alpha_{h} & h_{0} - 1
    s\theta_{h} c\alpha_{h} + s\alpha_{h} c\alpha_{h} & c\theta_{h} c\alpha_{h} + s\alpha_{h} c\alpha_{h} & -s\theta_{h} c\alpha_{h} + s\alpha_{h} c\alpha_{h} & 0
    s\theta_{h} & c\theta_{h} & 0 & 0
    0 & 0 & 0 & 1
\end{bmatrix}
\]

(25)

The forward kinematics model of the manipulator is found using the equation:

\[
q^T_T = q^T_1 T_2 T_3 T_4 T_5 .
\]

(26)

The same path generation problems previously discussed will be used here. The initial and final joints’ angles corresponding to the initial and final configurations of the end-effector along the Cartesian path are
Fig. 30. Joints' paths of 3R redundant planar manipulator for (a) first joint, (b) second joint, and (c) third joint.

Table 8

<table>
<thead>
<tr>
<th>Joint number (h)</th>
<th>$\alpha_{h-1}$ ($^\circ$)</th>
<th>$d_{h-1}$</th>
<th>$d_h$</th>
<th>$\theta_h$</th>
<th>$\theta_{lower}$ (h) ($^\circ$)</th>
<th>$\theta_{upper}$ (h) ($^\circ$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\theta_1$</td>
<td>-180</td>
<td>180</td>
</tr>
<tr>
<td>2</td>
<td>-90</td>
<td>0</td>
<td>0</td>
<td>$\theta_2$</td>
<td>-180</td>
<td>180</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.45</td>
<td>0.15</td>
<td>$\theta_3$</td>
<td>-180</td>
<td>180</td>
</tr>
<tr>
<td>4</td>
<td>-90</td>
<td>0.05</td>
<td>0.45</td>
<td>$\theta_4$</td>
<td>-180</td>
<td>180</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.15</td>
<td>0.05</td>
<td>$\theta_5$</td>
<td>-180</td>
<td>180</td>
</tr>
<tr>
<td>6</td>
<td>-90</td>
<td>0.15</td>
<td>0</td>
<td>$\theta_6$</td>
<td>-180</td>
<td>180</td>
</tr>
</tbody>
</table>

not given (free end points). The path point deviation and the accumulative deviation between the desired and generated paths are as given in Eqs. (22) and (23) where the path point deviations at the end points are included.

The evolutionary progress plot of the best-fitness individual using the deterministic mutation center and the knowledgeable dispersion factor and the path point deviation for the first path generation problem given in Eq. (18) are shown in Fig. 33. It is noted that the
Fig. 31. Effect of different variations in mutation center on the convergence speed of the algorithm for 3R redundant planar manipulator using (a) random dispersion factor, and (b) knowledgeable dispersion factor.

Fig. 32. Evolution of different path points of 3R redundant planar manipulator for (a) first joint, (b) second joint, and (c) third joint.
Fig. 33. (a) Evolutionary progress plot for the best-of-generation individual of general 6R manipulator using the first path generation problem, and (b) the corresponding path point deviation.

Fig. 34. Joints’ paths of the general 6R manipulator using the first path generation problem for (a) first joint, (b) second joint, and (c) third joint.
Fig. 35. Joints’ paths of the general 6R manipulator using the first path generation problem for (a) fourth joint, (b) fifth joint, and (c) sixth joint.

Fig. 36. (a) Evolutionary progress plot for the best-of-generation individual of general 6R manipulator using the second path generation problem, and (b) the corresponding path point deviation.
Fig. 37. Joints’ paths of the general 6R manipulator using the second path generation problem for (a) first joint, (b) second joint, and (c) third joint.

The algorithm converges within about 500 generations and the average path point deviation is almost 0.00057 m. The joints’ paths for the first three joints are shown in Fig. 34 while the joints’ paths for the last three joints are shown in Fig. 35. Regarding the second path generation problem given in Eq. (21), the evolutionary progress plot of the best-fitness individual using the deterministic mutation center and the knowledgeable dispersion factor and the path point deviation are shown in Fig. 36. It is noted that the algorithm converges within about 300 generations and the average path point deviation is almost 0.00055 m. The joints’ paths for the first three joints are shown in Fig. 37 while the joints’ paths for the last three joints are shown in Fig. 38. It is obvious that the resulting curves in joint space are smooth. In addition to that, it is clear that the generated paths are away from joints’ limits that are within the range \([-180^\circ, 180^\circ]\) for all joints.

Currently, we are exploring the effects of different genetic-algorithm related parameters and problem related parameters on the performance of the proposed algorithm. Furthermore, future versions of the algorithm will be developed that take other objective functions into consideration in addition to the accumulative path deviation in case of multiple or infinite solutions categories such as minimizing the net displacements of the joints, collision-free path planning, etc. The case in which both position and orientation are taken into account will also be considered later since it needs some special treatment. In addition to...
that, the trajectory generation problem will be covered in a later work of our research group.

5. Conclusions

In this paper, a framework tackling the problem of Cartesian path generation of robot manipulators is introduced. Central to our approach is the novel use of continuous genetic algorithms where smooth curves are used for representing the required geometric paths in the joint space. The proposed method emerges as a very flexible and competitive technique. It can be applied to any general serial manipulator with revolute and prismatic joints having positional degrees of freedom, which might not have a derived closed-form inverse kinematics solution. Furthermore, it can be applied to both nonredundant and redundant manipulators. The resulting joints’ paths are not only smooth but also provide a very high accuracy along the generated paths.

Three main categories of path generation problems were considered in this work depending on the number of solutions obtained: the unique solution category, the multiple solutions category and the infinite number of solutions category. The first two categories are applied in case of nonredundant manipulators while the last category is applied in case of redundant manipulators. In addition to that, we have classified the path generation problems into two types depending on whether the initial and final joints’ angles corresponding to the initial and final configurations of the end-effector are
given (fixed end points) or not (free end points). Each of the above mentioned categories was covered in details using the appropriate manipulators. It was noted that the toughest task to the continuous genetic algorithms is the multiple solutions category since the end points are free and the number of solutions is finite. The unique solution category comes in the second rank from toughness point of view while the easiest task to the algorithm is the infinite solutions category in case of redundant manipulators.

The effect of different variations of the mutation center and the dispersion factor used in the mutation operator on the convergence speed of the algorithm was investigated. The deterministic mutation center results in the fastest convergence speed among other mutation center schemes while the knowledgeable dispersion factor method has faster convergence speed than that of the random dispersion factor method for all of the previous categories. It was also noted that the initialization functions and the joints’ limits methods have minor effect on the convergence speed of the algorithm. However, the tangent hyperbolic function and the fixed limit method generally result in faster convergence speed than others. In addition to that, it was observed that the algorithm tries to find safe solutions that are away from joints’ limits.

Acknowledgements

The corresponding author would like to express his deep appreciation for the financial assistance provided by the Jordan Armed Forces as research fellowship. The continued support of Dr. Abdullah Sadiq, Rector PIEAS, is gratefully acknowledged. Sincere thanks to the PIEAS Computer Center for making their computer facilities available for this research work.

References


Za'er S. Abo-Hammour received the B.S. degree in Electrical Engineering from Mu’tah University in Karak, Jordan, in 1991, the M.S. in systems engineering from Pakistan Institute of Engineering and Applied Sciences, Pakistan, in 1998. He has been with the Jordan Armed Forces since 1991. Currently, he is pursuing a Ph.D. program in Pakistan Institute of Engineering and Applied Sciences in the field of continuous genetic algorithms and their applications in the motion planning of robot manipulators and in the numerical solution of boundary value problems. His research interests are focused on Control Systems, Optimal Control, Robotics, Numerical Methods and Artificial Intelligence Systems.

Nasir M. Mirza received his Master’s degree from the Physics Department of Quaid-i-Azam University, Islamabad in 1980 and got his Ph.D. in Computer Simulation of Diffusion of Neutral Particles Across Heterogeneous Interfaces from the Purdue University, USA in 1989. He is currently Head of Physics and Applied Mathematics Department at Pakistan Institute of Engineering and Applied Sciences. His research interests include Genetic Algorithms, Simulation and Modeling of Physical Systems, Application of Monte Carlo Methods, Discrete and Continuous System Simulations.
Sikander M. Mirza completed his Master’s degree in Physics from the Quaid-i-Azam University, Islamabad and earned his Ph.D. degree in Computational Studies in Transport Physics from Purdue University, USA in 1990. Since then, he has been working as full time faculty member at PIEAS. His research interests involve Computer Simulation of Dynamical Systems, Nonlinear systems, Modeling of Stochastic systems and Advanced Numerical Techniques.

Muhammad Arif has received B.E. in Electronics Engineering from NED University of Engineering and Technology, Karachi in 1990, then M.Sc. in Nuclear Engineering from CNS, Quaid-e-Azam University in 1993. He has received his Ph.D. in Information Sciences from Tohoku University, Japan in 1999 under S&T scholarship scheme. His research interests include Neural Networks, Genetic Algorithms, Linear and Nonlinear Control, Iterative Learning Control, Human Learning in Control, System identification, and Digital Image Processing. He is working in the Department of Informatics and Computer Science, Pakistan Institute of Engineering and Applied Sciences as Assistant Professor.