Trajectory planning of redundant manipulators using genetic algorithms

Maria da Graça Marcos, J.A. Tenreiro Machado, T.-P. Azevedo-Perdicoúlis

A B S T R A C T

The trajectory planning of redundant robots is an important area of research and efficient optimization algorithms are needed. This paper presents a new technique that combines the closed-loop pseudoinverse method with genetic algorithms. The results are compared with a genetic algorithm that adopts the direct kinematics. In both cases the trajectory planning is formulated as an optimization problem with constraints.

Keywords:
Redundant manipulators
Kinematics
Genetic algorithms
Trajectory planning

1. Introduction

Kinematic redundancy occurs when a manipulator possesses more degrees of freedom than the required to execute a given task. In this case the inverse kinematics admits an infinite number of solutions, and a criterion to select one of them is required. Most of the research on redundancy deals with the use of these extra degrees of freedom and is referred to in the literature as the resolution of redundancy [1].

Many techniques for solving the kinematics of redundant manipulators that have been suggested control the end-effector indirectly, through the rates at which the joints are driven, using the pseudoinverse of the Jacobian (see, for instance, [2]). The pseudoinverse of the Jacobian matrix guarantees an optimal reconstruction of the desired end-effector velocity – in the least-squares sense – with the minimum-norm joint velocity. However, even though the joint velocities are instantaneously minimized, there is no guarantee that the kinematic singularities are avoided [3]. Moreover, this method has the generally undesirable property that repetitive end-effector motions do not necessarily yield repetitive joint motions. Klein and Huang [4] were the first to observe this phenomenon for the case of the pseudoinverse control of a planar three-link manipulator.

Baillieul [5] proposed a modified jacobian matrix called the extended Jacobian matrix. The extended jacobian is a square matrix that contains the additional information necessary to optimize a certain function. The inverse kinematic solutions are obtained through the inverse of the extended jacobian. The algorithms, based on the computation of the extended jacobian

matrix, have a major advantage over the pseudoinverse techniques because they are locally cyclic [6]. The disadvantage of this approach is that, while mechanical singularities may be avoided, typical algorithmic singularities [7] arise from the way the constraint restricts the motion of the mechanism [8].

One optimization method that is gaining popularity for solving complex problems in robotics is the genetic algorithm (GA). GAs are population-based stochastic and global search methods. Their performance is superior to that revealed by classical techniques [9] and has been used successfully in robot path planning.

Parker et al. [10] used GAs to position the end-effector of a robot at a target location, while minimizing the largest joint displacement. This method has some shortcomings, such as the lack of location precision, and is affected by the values of the weights. Arakawa et al. [11] proposed a virus-evolutionary genetic algorithm, composed of a host population and a virus population with subpopulations, for the trajectory generation of redundant manipulators without collision, that optimize the total energy. The operators of crossover, mutation, virus infection and selection are executed in each subpopulation independently. Kubota et al. [12] studied a hierarchical trajectory planning method for a redundant manipulator using a virus-evolutionary GA. This method runs, simultaneously, two processes. One process calculates some manipulator collision-free positions and the other generates a collision-free trajectory by combining these intermediate positions. de la Cueva and Ramos [13] proposed a GA for planning paths without collisions for two robots, both redundant and non-redundant, sharing the same workspace. The GA works directly over the task space adopting the direct kinematics. Each robot is associated to one population and each string of a population represents a complete robot path. Nishimura et al. [14] proposed a motion planning method using an artificial potential field and a GA for a hyper-redundant manipulator whose workspace includes several obstacles. The motion planning is divided into two sub problems. The first is the "path planning" that generates a path leading the tip of manipulator to the goal without collisions, using the artificial potential field concept. The second consists in the "collision-free sequence generation" that generates a sequence of movements by which distinct parts of the manipulator can avoid collisions with the obstacles. McAvoy and Sangolola [15] proposed an approach with GAs for optimal point-to-point motion planning of kinematically redundant manipulators. Their approach combines B-spline curves, for the generation of smooth trajectories, with GAs for obtaining the optimal solution. Peng and Wei [16] presented the ASAGA trajectory planning method of redundant manipulators by combining a stochastic search algorithm (simulated annealing algorithm) and a GA. In the ASAGA the selection, crossover and mutation operators are adjusted by using an adaptive mechanism based on the fitness value. Zhang et al. [17] proposed an algorithm to solve the inverse kinematics of a flexible macro–micro manipulator system which combines a GA and a neural network. The GA is used to acquire discrete solution of the inverse kinematics, based on the fitness function and the discrete solution, is generalized through a forward neural network.

Having these ideas in mind, the paper is organized as follows. Section 2 introduces the fundamentals of the kinematics and dynamics of redundant manipulators. Based on these concepts, Section 3 presents the new closed-loop inverse kinematics algorithm with genetic algorithms (CLGA) and, for comparison purposes, the open-loop genetic algorithm (OLGA). Section 4 presents the simulation results and, finally, Section 5 draws the main conclusions.

2. Kinematics and dynamics of redundant manipulators

We consider a manipulator with n degrees of freedom whose joint variables are denoted by $q = [q_1, q_2, \ldots, q_n]^T$. We assume that the class of tasks we are interested in can be described by m variables, $\mathbf{x} = [x_1, x_2, \ldots, x_m]^T$, $m < n$, and that the relation between $\mathbf{q}$ and $\mathbf{x}$ is given by the direct kinematics:

$$\mathbf{x} = f(\mathbf{q}) \tag{1}$$

Differential kinematics of robot manipulators was introduced by Whitney [18] that proposed the use of differential relationships to solve for the joint motion from the Cartesian trajectory of the end-effector. Whitney named this method *resolved motion rate control*. Differentiating (1) with respect to time yields:

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2}$$

where $\dot{\mathbf{x}} \in \Re^m$, $\dot{\mathbf{q}} \in \Re^n$ and $\mathbf{J}(\mathbf{q}) = \partial f(\mathbf{q})/\partial \mathbf{q} \in \Re^{m \times n}$. Hence, it is possible to calculate a path $\mathbf{q}(t)$ in terms of a prescribed trajectory $\mathbf{x}(t)$ in the operational space.

Eq. (2) can be inverted to provide a solution in terms of the joint velocities:

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}(\mathbf{q})\dot{\mathbf{x}} \tag{3}$$

where $\mathbf{J}^{\#}$ is the Moore–Penrose generalized inverse of the Jacobian $\mathbf{J}$ [2,19].

The dynamic equation of motion for a general n-link manipulator can be described by

$$\mathbf{T} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) \tag{4}$$

where $\mathbf{T}$ is the $n \times 1$ joint torque vector, $\mathbf{M}(\mathbf{q})$ is the $n \times n$ inertia matrix, $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ is the $n \times 1$ Coriolis/centripetal vector, and $\mathbf{g}(\mathbf{q})$ is the $n \times 1$ gravity vector.

3. Robot trajectory control

The Jacobian of a n-link planar manipulator (i.e., $m = 2$) has a simple recursive nature according with the expressions:

$$\mathbf{J} = \begin{bmatrix} -l_1 S_1 - \cdots - l_n S_{1\ldots n} & \cdots & -l_n S_{1\ldots n} \\ l_1 C_1 + \cdots + l_n C_{1\ldots n} & \cdots & l_n C_{1\ldots n} \end{bmatrix} \tag{5}$$

where l_i is the length of link i, $q_{i\ldots k} = q_i + \cdots + q_k$, $S_{i\ldots k} = Sin(q_{i\ldots k})$ and $C_{i\ldots k} = Cos(q_{i\ldots k})$, $i, k \in \mathbb{N}$.

In the experiments are adopted arms having identical link lengths, $l_1 = l_2 = \ldots = l_n$.

In the closed-loop pseudoinverse (CLP) method the joint positions can be computed through the time integration of (6):

$$\Delta\mathbf{q} = \mathbf{J}^{\#}(\mathbf{q})(\mathbf{x}_{ref} - \mathbf{x}) \tag{6}$$

where $\mathbf{x}_{ref}$ is the vector of reference position in the operational space. Nevertheless, in a previous study, addressing the CLP method [20], we concluded that this method leads to unpredictable arm configurations and reveals properties resembling those that occur in chaotic systems.

Genetic algorithms (GAs) are a method for solving both constrained and unconstrained optimization problems, based on the mechanics of natural genetics and selection, that was first introduced by Holland [21]. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the fitness or the cost function. The GA modifies repeatedly the population of individuals (possible solutions). At each step, the genetic algorithm selects individuals at random, from the current population, to be parents, and uses them to produce the offspring for the next generation. Over successive generations, the population evolves towards an optimal solution. The GAs can be applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, not differentiable, stochastic, or highly nonlinear.

Bearing these facts in mind, in this paper we propose a new method that combines the CLP with a GA, that we call closed-loop inverse kinematics algorithm with genetic algorithms (CLGA).

For comparison purposes another GA is used to solve the redundant robot kinematics, adopting the direct kinematics to search the configuration space for the solution. We call this second scheme as the open-loop genetic algorithm (OLGA). The optimal configuration is the one that minimizes the fitness function according to some specified criteria.

3.1. The CLGA formulation

The CLGA adopts the closed-loop structure without requiring the calculation of the pseudoinverse. The CLGA uses an extended Jacobian matrix $\mathbf{J}^*$, $n \times n$, and an extended vector $\Delta\mathbf{x}^*$, as a way to limit the joint configurations for a given end-effector position.

The definition of $\mathbf{J}^*$ and $\Delta\mathbf{x}^*$ take the form:

$$\mathbf{J}^* = \begin{bmatrix} -l_1 S_1 - \cdots - l_n S_{1\ldots n} & \cdots & -l_n S_{1\ldots n} \\ l_1 C_1 + \cdots + l_n C_{1\ldots n} & \cdots & l_n C_{1\ldots n} \\ j_{31} & \cdots & j_{3n} \\ \cdots & \cdots & \cdots \\ j_{n1} & \cdots & j_{nn} \end{bmatrix} \quad \Delta\mathbf{x}^* = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \vdots \\ \Delta x_n \end{bmatrix} \tag{7}$$

where the matrix elements j_{ik} and Δx_i, $i = 3, \ldots, n$ and $k = 1, \ldots, n$, are values generated by the GA, satisfying the additional imposed constraints.

3.2. Representation and operators in the CLGA

An initial population of strings, with dimension $n_P = N$, is constructed at random and the search is then carried out among this population. Each chromosome (string) is implemented by a matrix of $n_V = (n-2) \times n$ values (genes), represented as floating-point numbers, and initialized in the range $[var_{min}, var_{max}]$. The three different operators used in the genetic algorithm are reproduction, crossover and mutation. In what respecting the reproduction operator, the successive generations of new strings are generated on the basis of their fitness function. In this case, it is used a rank weighting to select the strings from the old to the new population. For the crossover operator, the strings are randomly grouped together into pairs. Single crossover is then performed among pairs. Finally, for the mutation operator, one variable value is replaced with a new random one. The first chromosome is not mutated due to the adoption of elitism.

The CLGA procedure is shown in Fig. 1, where $\mathbf{x}_{ref}$ is the vector of reference position in the operational space and $\mathbf{x}_{ini}$ is a vector representing the initial position of the end-effecter in the operational space.

3.3. Optimization criteria

The fitness function is designed according to the goal we want to achieve. Five criteria have been selected *a priori*. All constraints and criteria are translated into penalty functions to be minimized and that are defined in the sequel.

Begin

$T = 0$

$\Delta \mathbf{x} = \mathbf{x}_{ref} - \mathbf{x}_{ini}$, $\mathbf{J}$

initialize random population $P(T) = \left[\left(\mathbf{J}^{(T,1)}, \Delta\mathbf{x}^{(T,1)}\right), \ldots, \left(\mathbf{J}^{(T,N)}, \Delta\mathbf{x}^{(T,N)}\right)\right]$ *where*

$\mathbf{J}^{(T,i)} = \left[\left(j_{31}^{(T,i)}, \ldots, j_{3n}^{(T,i)}\right), \ldots, \left(j_{n1}^{(T,i)}, \ldots, j_{nn}^{(T,i)}\right)\right]$ *and* $\Delta\mathbf{x}^{(T,i)} = \left[\Delta x_3^{(T,i)}, \ldots, \Delta x_n^{(T,i)}\right]$

get $\Delta\mathbf{q} = \mathbf{J}^{*-1}(\mathbf{q})\Delta\mathbf{x}^{*}$ *and* $\mathbf{q} = \int \Delta\mathbf{q}$

evaluate $P(T)$

Repeat

selection parents from $P(T)$

crossover $P(T)$

mutation $P(T)$

form new generation $P(T)$

get $\Delta\mathbf{q} = \mathbf{J}^{*-1}(\mathbf{q})\Delta\mathbf{x}^{*}$ *and* $\mathbf{q} = \int \Delta\mathbf{q}$

evaluate $P(T)$

$T = T + 1$

until *termination condition is TRUE*

get new $\mathbf{q}$

End

Fig. 1. Procedure for the CLGA.

1. The largest joint displacement between two adjacent robot configurations can be minimized through the fitness function:

$$f_1 = \max\{[q_j(k+1) - q_j(k)]^2\}, \quad j = 1,2,3 \tag{8}$$

where k and $k+1$ are two consecutive sampling instants.

2. The total level of joint velocities must be minimized at each configuration leading to the fitness function:

$$f_2 = \sum_{j=1}^{3} \dot{q}_i^2 \tag{9}$$

3. The joint accelerations are used to minimize the ripple in the time evolution of the robot trajectory, according to the fitness function:

$$f_3 = \sum_{j=1}^{3} \ddot{q}_i^2 \tag{10}$$

4. In order to minimize the total joint torque in each joint configuration the fitness function is

$$f_4 = \sum_{i=1}^{3} T_i^2 \tag{11}$$

5. To minimize the total joint power consumption the fitness function is

$$f_5 = \sum_{j=1}^{3} P_i^2 \tag{12}$$

where the power P_i at joint i is defined as $P_i = T_i\dot{q}_i$ where T_i is the generalized force/torque for joint i ($i = 1,2,3$).

3.4. The OLGA formulation

The OLGA trajectory planning adopts a simple open-loop structure, as we can see in Fig. 2. An initial population of strings, with dimension $n_P = N$, is constructed at random and the search is then carried out among this population. Each chromosome is defined through an array of $n_V = n$ values, q_i, $i = 1,\ldots,n$, represented as floating-point numbers, initialized in the range $[q_{\min}, q_{\max}]$. The end-effector position for each configuration is easily calculated using the direct kinematics.

Begin
  $T = 0$
  initialize random population $P(T) = \left[\left(q_1^{(T,1)}, \ldots, q_n^{(T,1)}\right), \ldots, \left(q_1^{(T,N)}, \ldots, q_n^{(T,N)}\right)\right]$
  get $\mathbf{X} = \left[\mathbf{x}^{(T,1)}, \ldots, \mathbf{x}^{(T,N)}\right]$ *using direct kinematics*
  evaluate $P(T)$
  Repeat
    selection parents from $P(T)$
    crossover $P(T)$
    mutation $P(T)$
    form new generation $P(T)$
    get $\mathbf{X}$ *using direct kinematics*
    evaluate $P(T)$
    $T = T + 1$
  until *termination condition is TRUE*
  get new $\mathbf{q}$
End

Fig. 2. Procedure for the OLGA.

If the robot's end-effector current position is $P_c = (x_c, y_c)$ and the desired final position is $P_f = (x_f, y_f)$, then the positional error, P_{error}, is defined as

$$P_{error} = \sqrt{(x_c - x_f)^2 + (y_c - y_f)^2} \tag{13}$$

The evaluation function $F_{\alpha,i}$ is defined based on the positional error, P_{error}, of the end-effector and on one of the criteria functions f_i ($i = 1, \ldots, 5$) defined previously:

$$F_{\alpha,i} = \alpha P_{error} + (1.0 - \alpha) f_i, \quad i = 1, \ldots, 5 \tag{14}$$

where $0 < \alpha < 1$ denotes a weighting factor. When $\alpha = 1.0$, the evaluation function is defined using only the position error, and in this case it would be defined by $F_{1.0,0}$.

4. Simulation results

This section presents the results of several simulations that compare the performance of the OLGA and the CLGA.

The experiments consist in the analysis of the kinematic performance of a planar manipulator with 3 rotational joints (3*R*-robot) that is required to repeat a circular motion in the operational space with frequency $\omega_0 = 7.0\ \mathrm{rad\,s^{-1}}$, center at $r = (x_1^2 + x_2^2)^{1/2}$, radius $\rho = 0.5$ and a step time increment of $\Delta t = 10^{-3}$ s. The goal here is to position the end-effector of the 3*R*-robot at a target location while satisfying a given optimization criterion.

4.1. The OLGA performance

In a first set of experiments, the OLGA adopts crossover and mutation probabilities of $p_c = 0.5$ and $p_m = 0.2$, respectively, a string population of $n_P = 100$, and the results are obtained for $n_G = 100$ consecutive generations.

Fig. 3 presents the average of the positional error, $\overline{P}_{error}$, for $n_C = 100$ cycles and $r = \{0.7, 1.0, 2.0\}$, defined as

$$\overline{P}_{error} = \frac{\sum_{i=1}^{k} P_{error}}{k} \tag{15}$$

where k is the number of sampling points and is defined as

$$k \approx \frac{2\pi}{\omega_0 \Delta t} n_C \tag{16}$$

As we can see the $\overline{P}_{error}$ is very large, revealing a considerable lack of precision. Therefore, we decided to develop a second set of experiments adopting a $n_P = 800$ string population in order to augment the search space. The results are also shown in Fig. 3. The algorithm requires more time, due to the higher computational load, but the performance, in what concerns the index $\overline{P}_{error}$, is much superior.

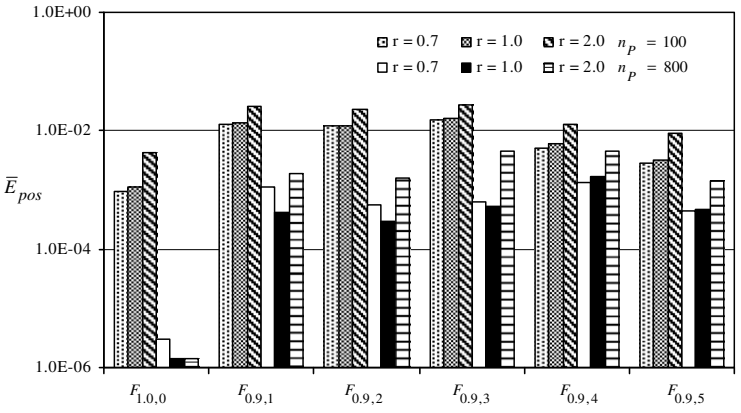

Fig. 3. $\overline{P}_{error}$ of the 3*R*-robot, under the action of the OLGA for $n_P = 100$ and $n_P = 800$ string population, during $n_C = 100$ cycles for $r = \{0.7, 1.0, 2.0\}$.

We observe that:

(i) as was expected, the minimum value of $\overline{P}_{error}$ occurs when function $F_{1.0,0}$ is used, that is, when we consider only the positional error on the fitness function;
(ii) the $\overline{P}_{error}$ depends upon r.

The Fourier transform of the robot joint velocities for a $n_P = 800$ string population is depicted in Figs. 4–6 revealing that:

(i) we get a signal energy distribution along all frequencies (corresponding to a chaotic response) when minimizing only the positional error $F_{1.0,0}$, minimizing the total joint torque $F_{0.9,5}$ or minimizing the total power consumption joint $F_{0.9,6}$;
(ii) the signal energy is essentially concentrated in the fundamental and multiple higher harmonics (corresponding to a repetitive motion) when minimizing the largest joint displacement $F_{0.9,1}$, the total joint level velocities $F_{0.9,3}$ or the joint accelerations $F_{0.9,4}$.

4.2. The CLGA performance

In this sub-section we start by analyzing the performance of the CLGA for a free workspace and, in a second phase, we study the effect of including several types of obstacles in the working environment.

The CLGA algorithm adopts crossover and mutation probabilities of $p_c = 0.5$ and $p_m = 0.2$, respectively, a $n_P = 100$ string population, and the results are obtained for $n_G = 100$ consecutive generations.

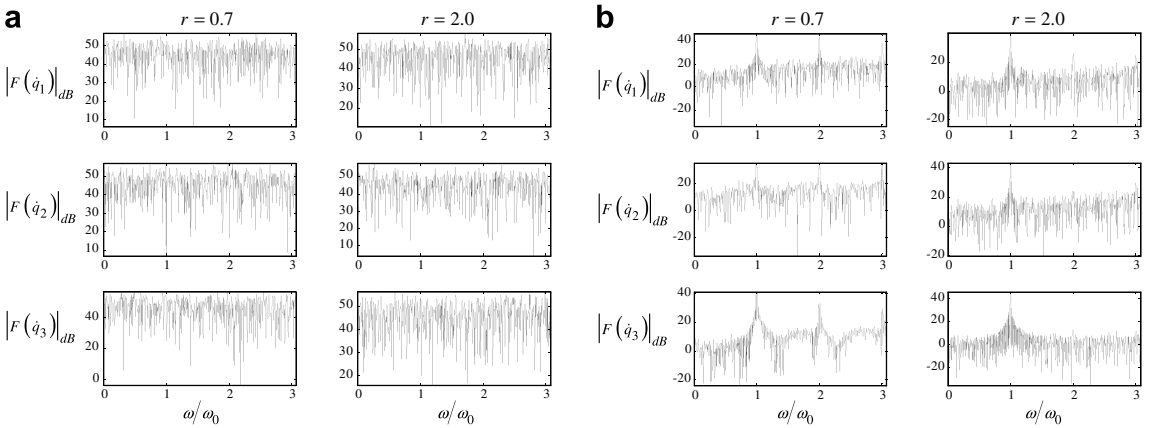

Fig. 4. $|F\{\dot{q}_i(t)\}|$ vs. ω/ω_0 of the 3*R*-robot, under the action of the OLGA, during $n_C = 100$ cycles for $r = \{0.7, 2.0\}$ and the fitnesses (a) $F_{1.0,0}$ and (b) $F_{0.9,1}$.

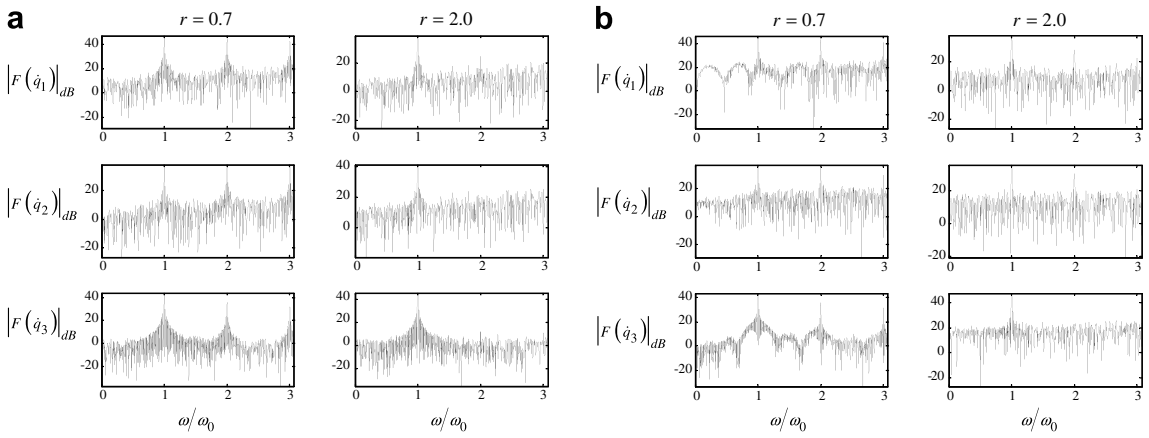

Fig. 5. $|F\{\dot{q}_i(t)\}|$ vs. ω/ω_0 of the 3*R*-robot, under the action of the OLGA, during $n_C = 100$ cycles for $r = \{0.7, 2.0\}$ and the fitnesses (a) $F_{0.9,2}$ and (b) $F_{0.9,3}$.

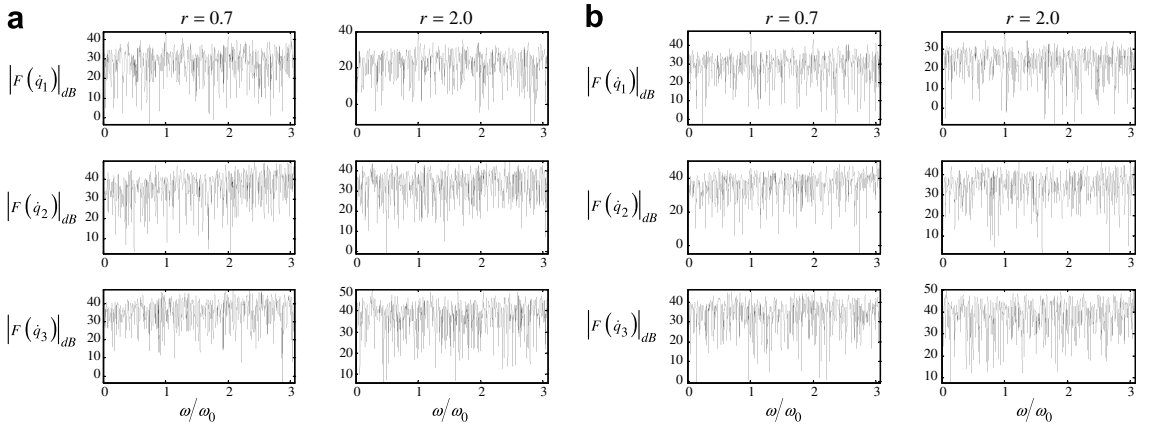

Fig. 6. $|F\{\dot{q}_i(t)\}|$ vs. ω/ω_0 of the 3*R*-robot, under the action of the OLGA, during $n_C = 100$ cycles for $r = \{0.7, 2.0\}$ and the fitnesses (a) $F_{0.9,4}$ and (b) $F_{0.9,5}$.

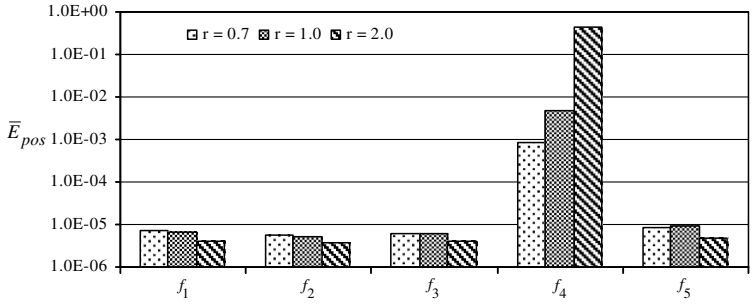

Fig. 7. $\overline{P}_{error}$ of the 3*R*-robot, under the action of the CLGA for $n_P = 100$ string population, during $n_C = 100$ cycles for $r = \{0.7, 1.0, 2.0\}$.

4.2.1. The CLGA performance in a workspace without obstacles

The average of the positional error, $\overline{P}_{error}$, is presented in Fig. 7. We observe that:

(i) a high precision is achieved when using the CLGA with $\{f_1, f_2, f_3, f_5\}$;
(ii) the maximum value of $\overline{P}_{error}$ occurs when minimizing the total joint torque f_4;
(iii) in general, the CLGA gives better accuracy in the positioning than the OLGA.

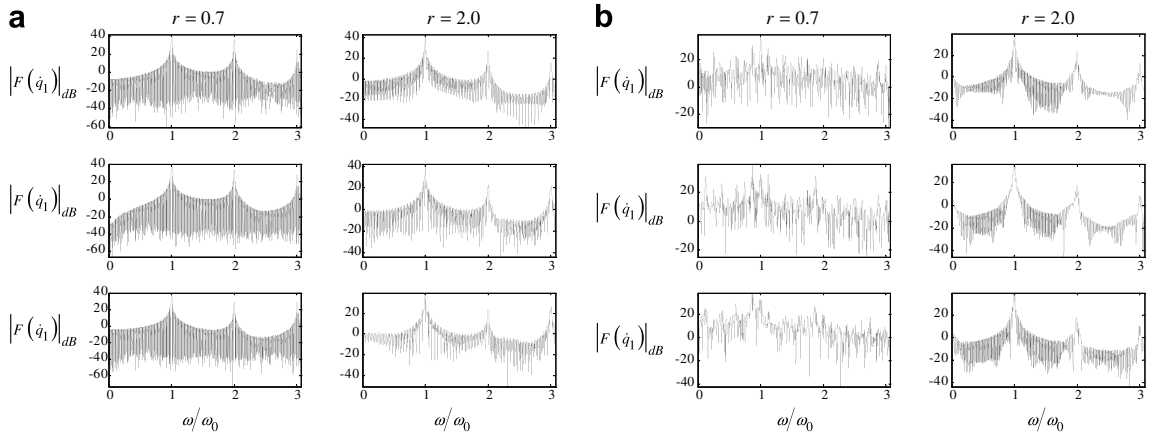

Fig. 8. $|F\{\dot{q}_i(t)\}|$ vs. ω/ω_0 of the 3*R*-robot, under the action of the CLGA, during $n_C = 100$ cycles for $r = \{0.7, 2.0\}$ and the fitnesses (a) f_1 and (b) f_2.

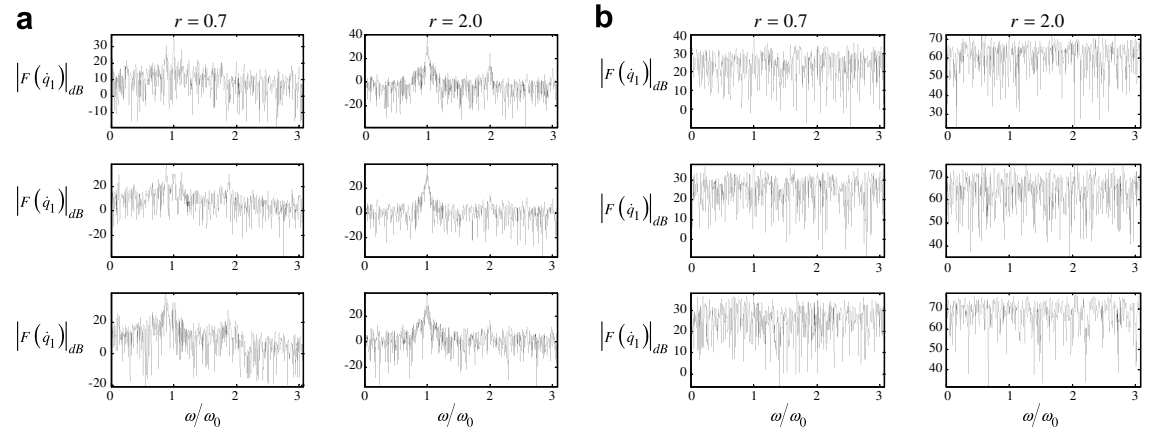

Fig. 9. $|F\{\dot{q}_i(t)\}|$ vs. ω/ω_0 of the 3*R*-robot, under the action of the CLGA, during $n_C = 100$ cycles for $r = \{0.7, 2.0\}$ and the fitnesses (a) f_3 and (b) f_4.

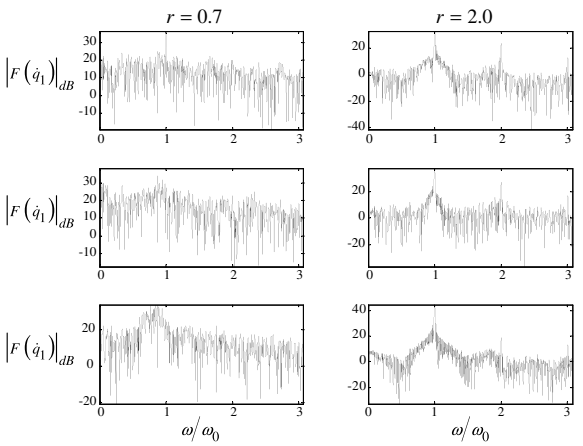

Fig. 10. $|F\{\dot{q}_i(t)\}|$ vs. ω/ω_0 of the 3*R*-robot, under the action of the CLGA, during $n_C = 100$ cycles for $r = \{0.7, 2.0\}$ and the fitness f_5.

The Fourier transform of the robot joint velocities are depicted in Figs. 8–10 revealing that:

(i) we get a signal energy distribution along all frequencies when minimizing the total joint torque f_4;
(ii) the signal energy is concentrated in the fundamental and multiple higher harmonics when we minimize the largest joint displacement f_1;

(iii) the results depend on the radial distance when minimizing the total joint velocities f_2, the total joint accelerations f_3 or the total joint power consumption f_5.

We verify that the CLGA has a better performance than the OLGA or the CLP because, for example, when minimizing the largest joint displacement f_1, we get not only a good positioning but also a repetitive trajectory.

4.2.2. The CLGA performance in a workspace with obstacles

This section presents the results of several simulations, for the criteria f_1, when considering two obstacles in the workspace. When, for a given joint configuration, some part of the manipulator is inside an obstacle, the CLGA simply rejects the configuration and generates a new population element.

Are considered two cases:

(i) for $r = 0.7$, the obstacles consist on one circle with center at (0.3, 1.3) and radius 0.2, and one rectangle, with the upper left corner and the lower right corner with coordinates (1.3, 0.9) and (1.8, 0.5), respectively;
(ii) for $r = 2.0$, the obstacles consist on one circle with center at (1.6, 0.6) and radius 0.2, and one rectangle, with the upper left corner and the lower right corner with coordinates (0.1, 1.5) and (0.6, 1.1), respectively.

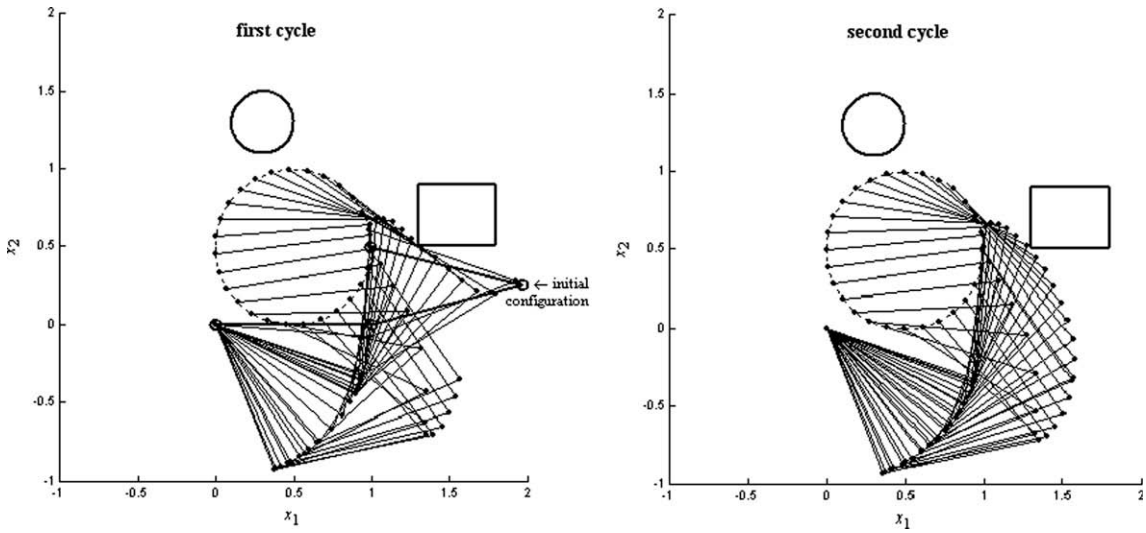

Fig. 11. Successive robot configurations in a workspace with obstacles for $r = 0.7$, for the first and second cycles, respectively.

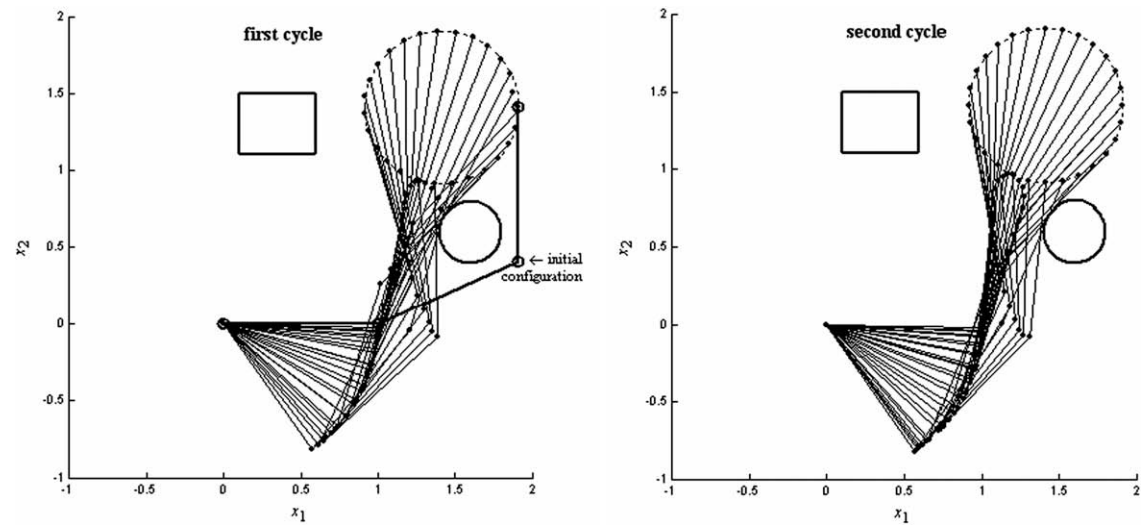

Fig. 12. Successive robot configurations in a workspace with obstacles for $r = 2.0$, for the first and second cycles, respectively.

In a first set of experiments we test the 3*R*-robot for a motion with two cycles.

For $r = 0.7$, the initial joint configuration is $q_0 = (0°, 14.9°, 154.3°)$. Fig. 11 show the successive robot configurations during the first and second cycles, respectively. We observe that the robot approaches the desired position while avoiding the obstacles, for the two cycles.

For $r = 2.0$, the initial joint configuration is $q_0 = (0°, 24.5°, 65.3°)$. Fig. 12 show the successive robot configurations during the first and second cycles, respectively. We observe that the robot can not reach some points in the circle for the first cycle but, for the second cycle, do not occur any problems. Due to this results, we repeat the experiment for $r = 2.0$, with the initial joint configuration $q_0 = (-16.5°, 90°, -25.8°)$. The results for the first cycle are shown in Fig. 13, revealing that the performance of the CLGA depends on the initial configuration of the manipulator.

In a second set of experiments we test the 3*R*-robot for $n_C = 100$ cycles. The initial joint configuration is $q_0 = (0°, 14.9°, 154.3°)$ for $r = 0.7$ and $q_0 = (-16.5°, 90°, -25.8°)$ for $r = 2.0$, leading to the average of the positional error, $\overline{P}_{error} = 6.55E-06$ and $\overline{P}_{error} = 9.44E-06$, respectively. The Fourier spectra of the joint velocities is depicted in Fig. 14.

The results reveal that the average of the positional error, $\overline{P}_{error}$, and the Fourier transform of the robot joint velocities, are consistent with those of the previous section. The presence of obstacles may cause some problems if the initial joint configurations is not adequate for the required task, namely in what repeatability and positioning is concerned. As we can see in Fig. 15, the presence of obstacles leads to unpredictable and severe variations in the joint positions, causing a high positional error. However, this problem disappears when we choose initial joint configurations more adequate for the desired trajectory and for the desired workspace, as we can see in Fig. 13.

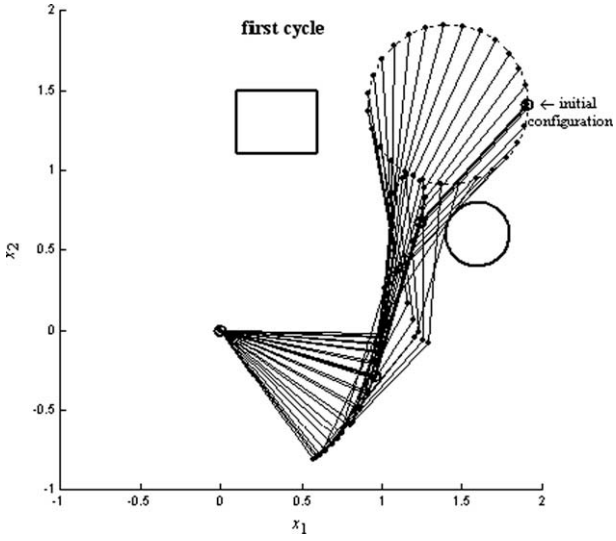

Fig. 13. Successive robot configurations in a workspace with obstacles for $r = 2.0$. The first and second cycles are identical.

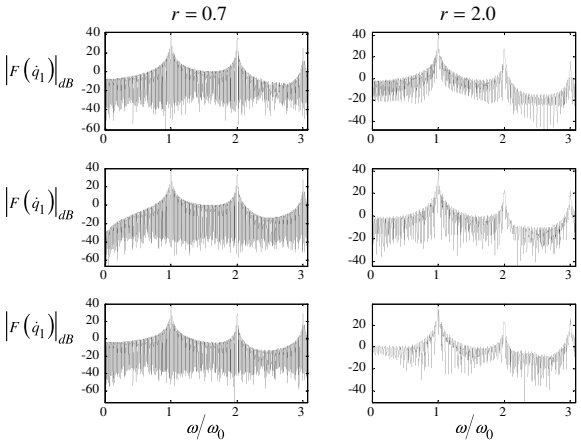

Fig. 14. $|F\{\dot{q}_i(t)\}|$ vs. ω/ω_0 of the 3*R*-robot, under the action of the CLGA, during $n_C = 100$ cycles for $r = \{0.7, 2.0\}$ in a workspace with obstacles.

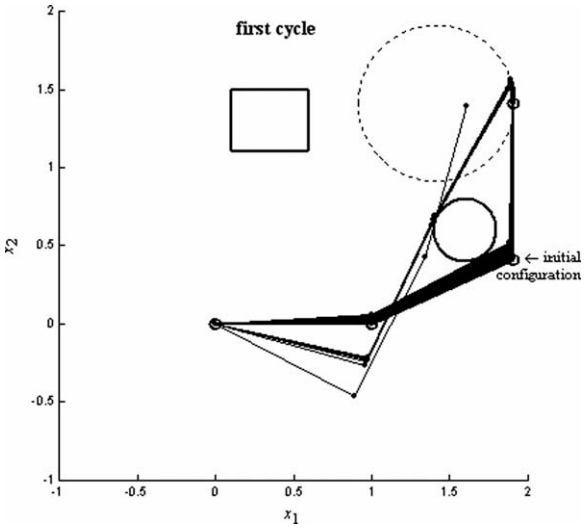

Fig. 15. Successive robot configurations in a workspace with obstacles for $r = 2.0$ and for the initial joint configurations $q_0 = (0°, 24.5°, 65.3°)$.

5. Conclusions

A CLGA scheme that combines the CLP with a GA scheme was presented. Several experiments were developed to study the performance of the CLGA when the manipulator is required to repeat a circular motion in the operational space while satisfying different optimization criteria. The results were compared with the standard algorithm OLGA that adopts the direct kinematics.

The results show that the CLGA gives, in general, superior results in what concerns the repeatability and positioning. The better result occurs when the CLGA minimizes the largest joint displacement between two adjacent configurations since not only we get a good positioning, but also the joint motion is repetitive and the chaotic phenomena observed in the CLP disappear. It is shown that the presence of obstacles does not present an additional complexity for the CLGA to reach the solution, as long as the selected initial joint configuration are adequate for the required task.

References

[1] Dragomir N Nenchev, Yuichi Tsumaki. Motion analysis of a kinematically redundant seven-DOF manipulator under the singularity-consistent method. In: Proceedings of the 2003 IEEE international conference on robotics and automation; 2003. p. 2760–5.
[2] Doty Keith L, Melchiorri C, Bonivento C. A theory of generalized inverses applied to robotics. Int J Robot Res 1993;12:1–19.
[3] Baillieul J, Hollerbach J, Brockett R. Programming and control of kinematically redundant manipulators. In: Proceedings of the 23rd IEEE conference on decision and control; 1984. p. 768–74.
[4] Klein CA, Huang CC. Review of pseudoinverse control for use with kinematically redundant manipulators. IEEE Trans Syst Man Cyber 1983;13: 245–50.
[5] Baillieul J. Kinematic programming alternatives for redundant manipulators. In: Proceedings of the IEEE international conference on robotics and automation; 1985. p. 722–8.
[6] Baker DR, Wampler II CW. On the inverse kinematics of redundant manipulators. Int J Robot Res 1988;7(2):3–21.
[7] Park Ki-Cheol, Chang Pyung-Hun, Lee Sukhan. A new kind of singularity in redundant manipulation: semi algorithmic singularity. Proc IEEE Int Conf Robot Autom 2002;2:1979–84.
[8] Park J, Chung W-K, Youm Y. Characteristics of optimal solutions in kinematics resolutions of redundancy. IEEE Trans Robot Autom 1996;12(3): 471–8.
[9] Goldenberg DE. Genetic algorithms in search optimization, and machine learning. Reading, MA: Addison-Wesley; 1989.
[10] Parker JK, Khoogar AR, Goldberg DE. Inverse kinematics of redundant robots using genetic algorithms. In: Proceedings of the 1989 IEEE international conference on robotics and automation; 1989. p. 271–6.
[11] Arakawa T, Kubota N, Fukuda T. Virus-evolutionary genetic algorithm with subpopulations: application to trajectory generation of redundant manipulator through energy optimization. In: Proceedings of the 1996 IEEE international conference on systems, man, and cybernetics; 1996. p. 14–7.
[12] Kubota N, Arakawa T, Fukuda T, Shimojima K. Trajectory generation for redundant manipulator using virus evolutionary genetic algorithm. In: Proceedings of the IEEE international conference on robotics and automation; 1997. p. 205–10.
[13] de la Cueva V, Ramos F. Cooperative genetic algorithms: a new approach to solve the path planning problem for cooperative robotic manipulators sharing the same work space. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems; 1998. p. 267–72.
[14] Nishimura T, Sugawara K, Yoshihara I, Abe K. A motion planning method for a hyper multi-joint manipulator using genetic algorithm. In: Proceedings of the IEEE international conference on systems, man, and cybernetics; 1999. p. 645–50.
[15] McAvoy B, Sangolola B. Optimal trajectory generation for redundant planar manipulators. In: Proceedings of the IEEE international conference on systems, man, and cybernetics; 2000. p. 3241–6.
[16] Peng Y, Wei W. A new trajectory planning method of redundant manipulator based on adaptive simulated annealing genetic algorithm (ASAGA). In: Proceedings of the IEEE international conference on computational intelligence and security; 2006. p. 262–5.
[17] Zhang Y, Sun Z, Yang T. Optimal motion generation of a flexible macro–micro manipulator system using genetic algorithm and neural network. In: Proceedings of the 2006 IEEE conference on robotics, automation and mechatronics; 2006. p. 1–6.

[18] Whitney DE. Resolved motion rate control of manipulators and human prostheses. IEEE Trans Man–Machine Syst 1969;MMS-10(2):47–53.
[19] Siciliano Bruno. Kinematic control of redundant robot manipulators: a tutorial. J Intell Robot Syst 1990;3:201–12.
[20] Marcos MG, Duarte FB, Machado JAT. Complex dynamics in the trajectory control of redundant manipulators. Trans Nonlinear Sci Complex 2006:134–43.
[21] Holland JH. Adaptation in natural and artificial systems. 2nd ed. Ann Arbor: University of Michigan Press, Mit Press; 1992.