

Congress on Evolutionary  
**COMPUTATION**  
2000





# A *GA* Perspective of the Energy Requirements for Manipulators Maneuvering in a Workspace with Obstacles

**E. J. Solteiro Pires**

Universidade Trás-os-Montes e Alto Douro  
Secção de Engenharias  
5001 Vila Real, Portugal  
epires@utad.pt

**J. A. Tenreiro Machado**

Instituto Superior de Engenharia do Porto  
Departamento de Engenharia Electrotécnica  
R. Antonio Bernardino de Almeida  
4200-072 Porto, Portugal  
jtm@dee.isep.ipp.pt

**Abstract - This paper proposes a genetic algorithm to generate trajectories for robotic manipulators. The objective is to minimize the ripple in the trajectory time evolution and to minimize the actuator energy requirements without colliding with any obstacles in the workspace. The article presents the results for several redundant and hyper-redundant manipulators.**

**Keywords:** Genetic algorithms, Robotics, Trajectory planning, Optimization.

## 1 Introduction

In the last decade genetic algorithms (*GAs*) have been applied in a plethora of fields such as in control, parameter and system identification, robotics, planning and scheduling, image processing, pattern recognition, speech recognition. This paper addresses the area of robotics, namely the trajectory planning for mechanical manipulators. Planning a robot trajectory consists in finding a continuous motion that takes the arm from a given starting configuration, without collision with any obstacle, up to a desired end position in the workspace.

Various methods for trajectory planning and collision avoidance schemes based on *GAs* have been proposed. A possible approach consists in adopting the differential inverse kinematics, using the Jacobian matrix, for generating the manipulator trajectories (Chen and Zalzalá, 1997; Davidor 1991). However, the algorithm must take into account the problem of kinematic singularities that may be hard to tackle. To avoid this problem, other algorithms for the trajectory generation are based on the direct kinematics (Doyle and Jones, 1996; Rana and Zalzalá, 1996; Kubota *et al.* 1997; Wang and Zalzalá 1996).

Chen and Zalzalá (1997) propose a *GA* method to generate the position and the configuration of a mobile manipulator. The authors study the optimization of the least torque norm, the manipulability, the torque distribution and the obstacle avoidance, through the inverse kinematics scheme.

Davidor (1991) also applies *GAs* to the trajectory generation by searching the inverse kinematics solutions to pre-defined end-effector robot paths.

Kubota *et al.* (1997) study 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.

Rana and Zalzalá (1996) developed a method to plan a near time-optimal, collision-free, motion in the case of multi-arm manipulators. The planning is carried out in the joint space and the path is represented as a *string* of via-points connected through cubic splines.

Doyle and Jones (1996) propose a path planning scheme that uses a *GA* to search the manipulator configuration space for the optimum path. The *GA* generates good path solutions but it is not sufficiently robust.

In this line of thought, this paper proposes a path planning method based on a *GA* while adopting the direct kinematics and the inverse dynamics. The optimal trajectory is the one that minimizes both the path length, the ripple in the time evolution and the energy requirements, without any collision with the obstacles in the workspace.

Bearing these facts in mind, this paper is organized as follows. Section 2 introduces the problem and the *GA*-based method for its resolution. Sections 3, 4 and 5 describe the solution representation, the *GA* operators used in trajectory planning and the optimization criteria, respectively. Based on this formulation, section 6 presents the results for several simulations involving different robot structures and obstacles in the workspace. Finally, section 7 outlines the main conclusions.

## 2 Problem formulation

In the study we consider robotic manipulators that are required to move from a initial configuration up to a given final point. In the experiments we consider 2, 3 and 4 link planar manipulators with rotational joints. The arms have identical link lengths and weights, making a total reach distance of 2 *m* and a total gross weight of 2 kg, and the robot joints are free to rotate 360°. Therefore, the manipulator workspace is a circle with 2 *m* radius, that may present obstacles such as rectangles and circles. To test a possible collision between the manipulator and the obstacles, the arm structure is discretized in several points and then these points are checked in order to verify if they are inside any obstacle.

In what concerns the trajectory generator, it is adopted a *GA* scheme to search for a global optimal robot path. The

manipulator trajectory consists in a set of strings that represent the joint positions between the initial and final robot configurations.

### 3 Representation

The path is encoded, directly, as strings in the joint space to be used by the *GA* as:

$$[(q_{11}, \dots, q_{k1}), \dots, (q_{1j}, \dots, q_{kj}), \dots, (q_{1n}, \dots, q_{kn})] \quad (1)$$

The  $i$ th joint variable for a robot intermediate  $j$ th position is  $q_{ij}$ , the chromosome is constituted by  $n$  genes (configurations) and each gene is formed by  $k$  values, where  $k$  is the number of robot links. The value of  $q_{ij}$  is represented as a floating-point number, that is initialized in the range  $[-360^\circ, +360^\circ]$ . This range is used to eliminate possible discontinuities that may occur when, for example, one joint angle attempts to pass its limits. It should be noted that the initial configuration has not been encoded into the string because this configuration remains unchanged throughout the search.

For simplicity, the normalized time between two consecutive configurations is considered  $\Delta t = 1$  sec, without losing generality, because it is always possible to perform a time re-scaling.

### 4 Operators in the genetic algorithm

An initial population of strings is constructed by generating random paths between the starting configuration and one final robot position. The search is then carried out among this population. The three different operators used in the genetic planning are reproduction, crossover and mutation, as described in the sequel.

In what concern the reproduction operator, the successive generations of new strings are reproduced on the basis of their fitness function. In this case, it is used a tournament selection (Goldberg 1989) to select the strings from the old population, up to the new population.

For the crossover operator, the strings in the new population are grouped together into pairs at random. Single crossover is then performed among pairs. The crossover point is only allowed between genes (*i.e.* the crossover operator may not disrupt genes).

Finally, for the mutation operator one gene value is replaced with a given probability and follows the equation:

$$q_{ij}(t+1) = q_{ij}(t) + k_m x \quad (2a)$$

$$x \sim U[-1; 1] \quad (2b)$$

where  $q_{ij}$  is the  $i$  value of  $j$  gene at generation  $t$ ,  $x$  is a uniform random number between  $[-1, 1]$  and  $k_m$  a parameter.

## 5 Evolution criteria

Six criteria have been selected to qualify the evolving manipulator paths. All constraints and criteria are translated into penalty functions to be minimized. Each criterion is computed individually and then, is used in the fitness function evaluation (Pires and Machado 1999).

The fitness function  $f$ , adopted to evaluate the candidate trajectories is defined as:

$$f = \begin{cases} -\alpha_1 \dot{q} - \alpha_2 \ddot{q} - \alpha_3 \dot{p} - \alpha_4 \ddot{p} - \alpha_5 \varepsilon - \alpha_6 P & nap = 0 \\ +\infty & nap \neq 0 \end{cases} \quad (3)$$

where  $\dot{q}$ ,  $\ddot{q}$ ,  $\dot{p}$ ,  $\ddot{p}$ ,  $\varepsilon$ ,  $P$  and  $nap$  are the criteria defined in the sequel. The optimization goal consists in finding a set of design parameters that minimize  $f$  according to the priorities given by the values of  $\alpha_i$  ( $i = 1, \dots, 6$ ).

The joint velocities  $\dot{q}$  are used to minimize the manipulator traveling distance. This criteria is defined as:

$$\dot{q} = \sum_{j=1}^n \sum_{i=1}^k \dot{q}_{ij}^2 \quad (4)$$

where  $\dot{q}_{ij}$  is the  $j$ th intermediate incremental position node of the  $i$ th joint and  $n$  is the number of the intermediate position in the simulation of the  $k$ -link manipulator. This equation is used to minimize the traveling distance because if the curve length is minimized, then the ripple in the space trajectory is indirectly reduced. For a function  $g(x)$  the distance curve length is  $\int [1 + (g')^2] dx$  and, consequently, to minimize the distance curve length it is adopted the expression  $\int (g')^2 dx$ . The fitness function maintains the quadratic terms so that the robot configurations are uniformly distributed between the initial and final configurations.

The joint accelerations  $\ddot{q}$  are used to minimize the ripple in the time evolution of the robot trajectory. This criteria is calculated as:

$$\ddot{q} = \sum_{j=1}^n \sum_{i=1}^k \ddot{q}_{ij}^2 \quad (5)$$

where  $k$ ,  $n$ ,  $i$  and  $j$  are defined as previously.

The cartesian velocities  $\dot{p}$  is introduced in the fitness function  $f$  to minimize the total trajectory length, from the initial point up to the final point. This criteria is defined as:

$$\dot{p} = \sum_{w=2}^n d(p_w, p_{w-1})^2 \quad (6)$$

where  $p_w$  is the robot  $w$  intermediate arm cartesian position and  $d(\cdot, \cdot)$  is a function that gives the distance between the two arguments.

The cartesian acceleration  $\ddot{p}$  in the fitness functions is responsible for reducing the ripple in time evolution of the arm velocities. This criteria is formulated as:

$$\ddot{p} = \sum_{w=3}^n |d(p_w, p_{w-1}) - d(p_{w-1}, p_{w-2})|^2 \quad (7)$$

where  $p_w$  and  $d(\cdot, \cdot)$  are defined as previously.

The end point distance  $\epsilon$  is a criterion that measures the distance between the desired end point and the end point reached in the simulation.

A trajectory planning algorithm must minimize the manipulator energy consumption. Two criteria are used independently to evaluate the power consumption  $P$ .

One criteria is the average of the absolute mechanical energy  $E_a$  during the total trajectory time  $T$  (Silva and Machado 1999). It is computed assuming that power regeneration is not available by motors doing negative work, that is, by taking the absolute value of the power. Therefore, the power consumption is calculated by the formula:

$$P_a = \frac{1}{T} E_a = \frac{1}{T} \sum_{j=1}^n \sum_{i=1}^k |\tau_j \cdot \theta_{ji}| \quad (8)$$

Another criteria is the average of the energy  $E_l$  dissipated by the electric motors (e.g. in the winding resistance) of the manipulator, that is captured by the expression:

$$P_l = \frac{1}{T} E_l = \frac{1}{T} \sum_{j=1}^n \sum_{i=1}^k \tau_j^2 \quad (9)$$

The points that are not admissible give a conflict measure between the robot and the obstacles. In this perspective, the  $nap$  value is evaluated as follows: each manipulator link is divided in  $p$  equal parts ( $p = (4, 3, 2)$  for the 2, 3 and 4 link manipulators). Therefore, the  $nap$  value is a criterion consisting on the sum of the manipulator points that are inside the obstacles.

## 6 Simulation results

This section presents the results of several simulations. The experiments consist on moving a robotic arm from the starting point  $A \equiv (1, 1)$  up to the final point  $B \equiv (-0.6697, 1.6168)$ . The initial configuration of the 2R robot is  $(q_1, q_2) = (0, 90^\circ)$ . The 3R and 4R robots adopt the configurations  $(q_1, q_2, q_3) = (10.9^\circ, 34.1^\circ, 34.1^\circ)$  and  $(q_1, q_2, q_3, q_4) = (0^\circ, 0^\circ, 90^\circ, 0^\circ)$ , respectively, because they have a 'geometric resemblance' that makes easier the result comparison. Moreover, the simulations are divided into two groups: workspace without obstacles and workspace with obstacles.

The algorithm adopts crossover and mutation probabilities of  $p_c = 0.8$  and  $p_m = 0.1$  respectively,  $k_m = 1.8$

and a 100-string population. For the experiment are used string lengths of  $l = 16$  and the selection operator is based on tournament selection with elitism.

The 2R and 3R robots are firstly tested in a workspace without obstacles and afterwards with one and two obstacles for all power criteria.

### 6.1. Workspace without obstacles

The 2R and 3R robots (Figs 1-4 and Figs 5-7) are firstly tested for the criteria  $P_a$  and  $\alpha = (1/k, 0.5, 1, 1, 20, 0.16)$ .

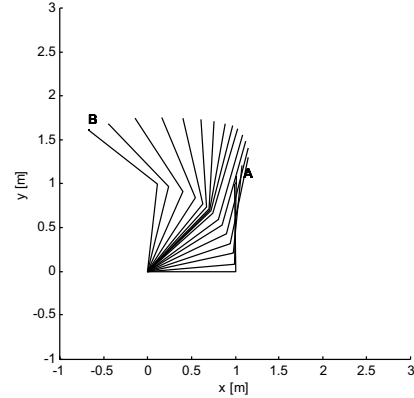


Figure 1: Successive configurations for a trajectory of the 2R robot and a workspace without obstacles.

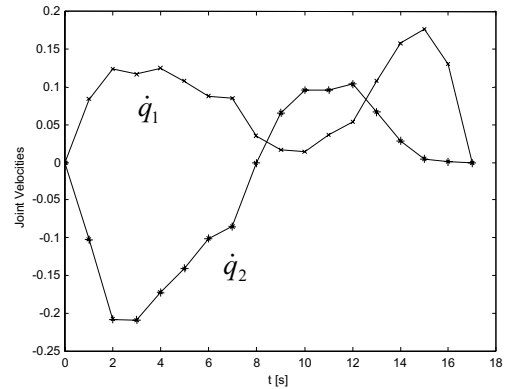


Figure 2: Joint velocities versus time for the 2R robot.

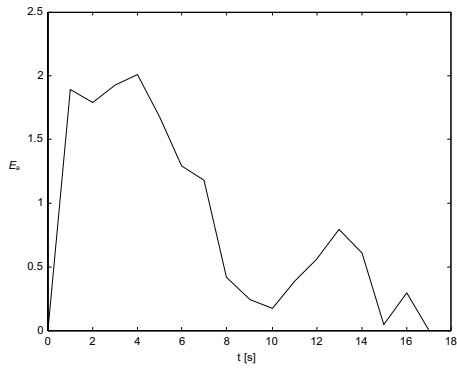


Figure 3:  $E_a(t)$  for the 2R robot.

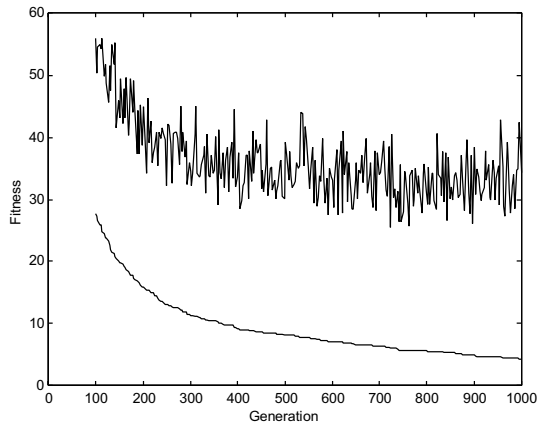


Figure 4: The best individual evolution and the fitness mean evolution versus generation for the 2R robot.

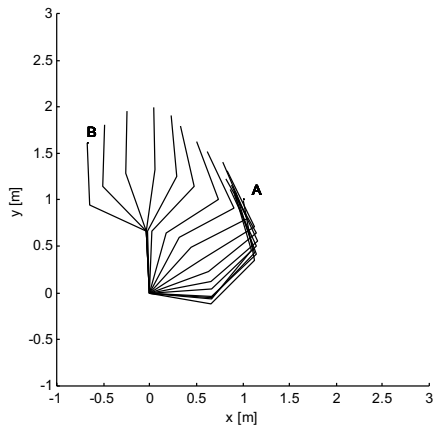


Figure 5: Successive configurations for a trajectory of the 3R robot and a workspace without obstacles.

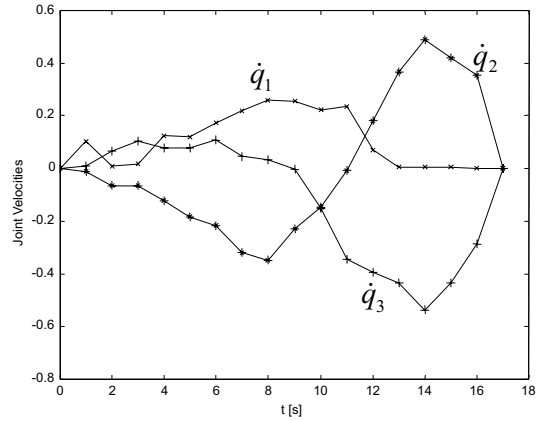


Figure 6: Joint velocities *versus* time for the 3R robot.

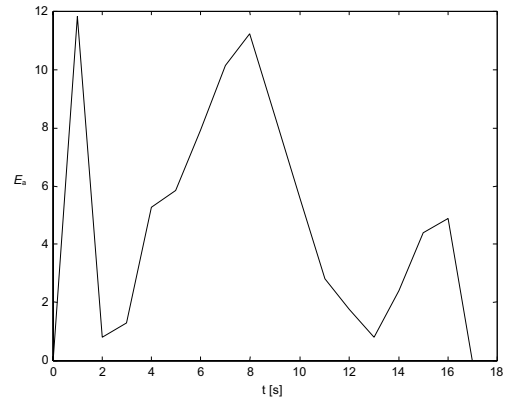


Figure 7:  $E_a(t)$  for the 3R robot.

The total ‘absolute’ energies required by the 2R, 3R and 4R manipulators to reach the final point are 15.3 J, 85.2 J and 118.6 J, respectively.

The results are satisfactory because the robot approaches the desired position without trajectory ‘oscillations’ and the time evolution of the variables presents a small ripple.

## 6.2. Workspace with obstacles

This section presents the manipulator trajectories for one and two obstacles in the workspace. The obstacles consist on one rectangle, with the upper left corner and the lower right corner with co-ordinates (0.3, 0.2) and (0.75, 0.8), respectively, and one circle, with center at (-0.7, -0.7) and radius 0.6.

The results for a 2R manipulator and one obstacle are shown in Figs 8 to 10 with  $\alpha = (1/k, 1/k, 1, 1, 20, 0.16)$ . For a workspace with two obstacles the 2R robot can not reach the goal point because it is not physically possible to pass between the obstacles.

For example, the results for a 4R robot and one obstacle workspace are shown in Figs 11 and 12.

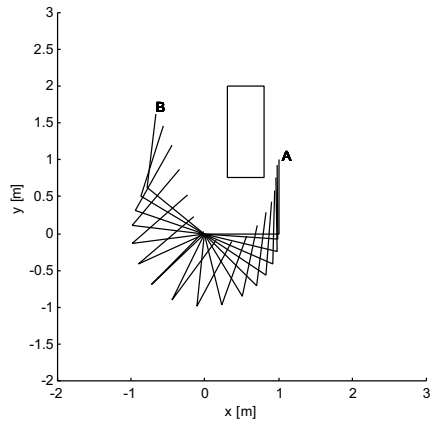


Figure 8: Successive configurations for a trajectory of the 2R robot and a workspace with a rectangular obstacle.

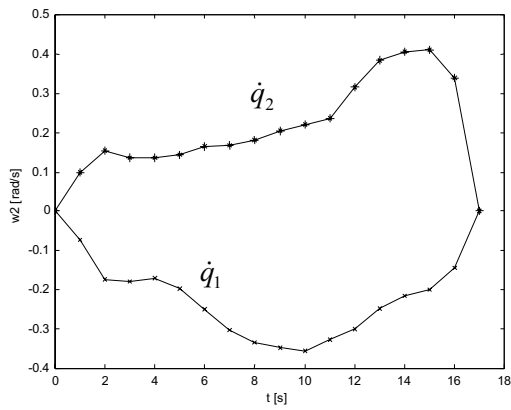


Figure 9: Joint velocities *versus* time for the 2R robot

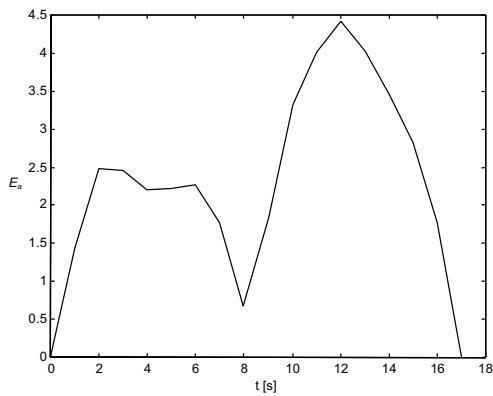


Figure 10:  $E_a(t)$  for the 2R robot

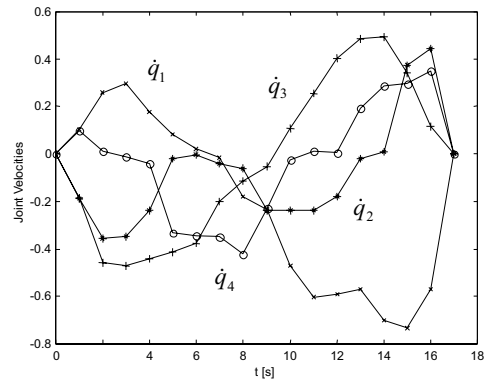


Figure 11: Joint velocities *versus* time for the 4R robot

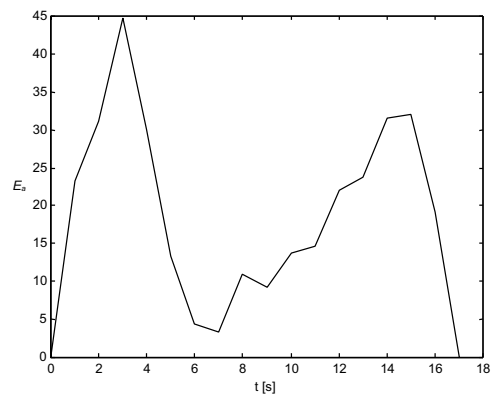


Figure 12:  $E_a(t)$  for the 4R robot

The total ‘absolute’ energies required for the 2R, 3R and 4R manipulators to reach the end points are 41.2 J, 116.6 J and 327.2 J, respectively, which reveals a high dependence with the number of robot dof.

For the 3R and 4R robots and a workspace with two obstacles, the results are shown in Figs. 13-16 for  $\alpha = (1/k, 1/k, 2, 2, 20, 0.16)$ . The absolute energies are 119.5 J and 259.2 J, respectively.

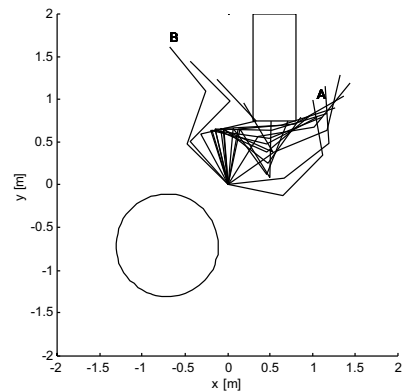


Figure 13: Successive configurations for a trajectory of the 3R robot and a workspace with two obstacles.

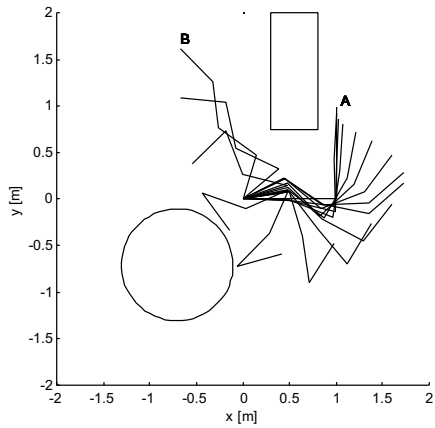


Figure 14: Successive configurations for a trajectory of the 4R robot and a workspace with two obstacles.

fact, figures 19-20 show that the energy dissipated required by the robots increases with the number of dof and the number of workspace obstacles.

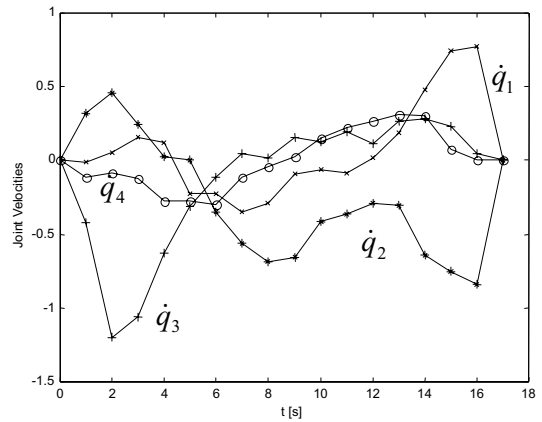


Figure 17: Joint velocities versus time for the 4R robot

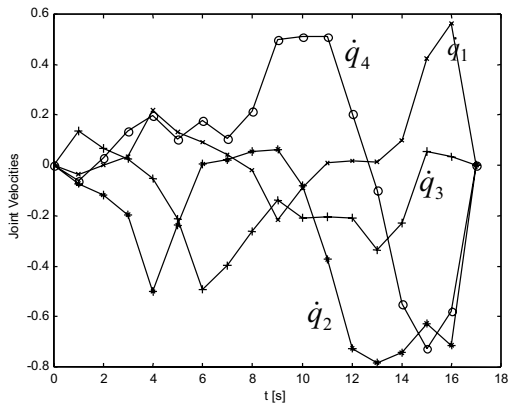


Figure 15: Joint velocities versus time for the 4R robot

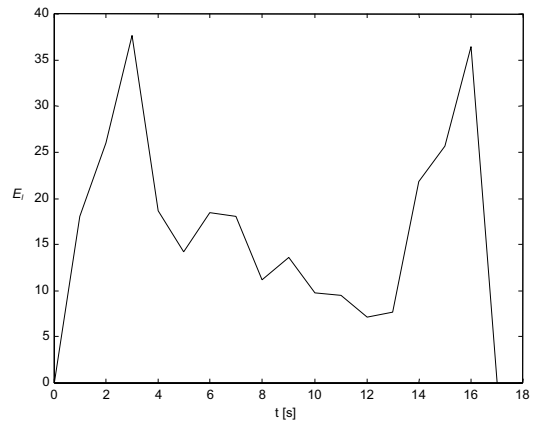


Figure 18:  $E_i(t)$  for the 4R robot.

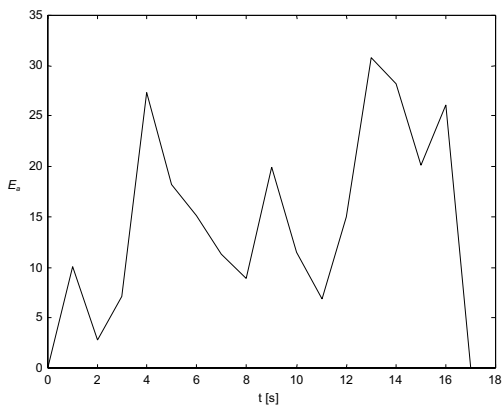


Figure 16:  $E_c(t)$  for the 4R robot.

In a second phase we test the power criteria  $P_i$ . For the 4-R robot, one obstacle and  $\alpha = (1/k, 0.5, 1, 1, 20, 0.0001)$  the results are similar to the previous ones (figures 17-18). In

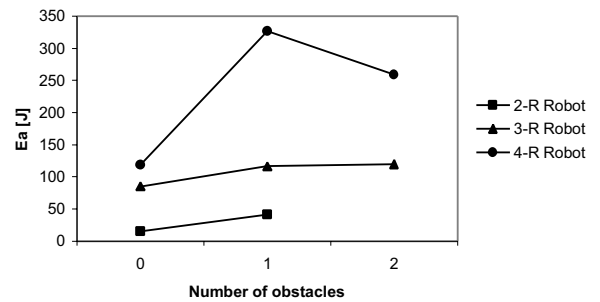


Figure 19: Energy  $E_a$  versus number of obstacles.



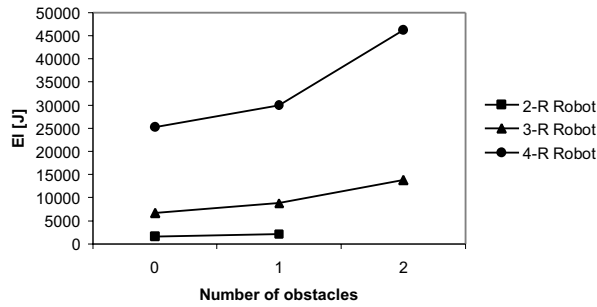


Figure 20: Energy  $E_l$  versus number of obstacles.

Robot	Without obstacles		With two obstacles	
	Optimization criteria			
	$P_a$	$P_l$	$P_a$	$P_l$
2R	15.3	13.6		
3R	85.2	62.0	119.5	77.9
4R	118.6	154.3	259.2	171.2

Table 1: Values of  $E_a$  for the experiments with zero and two obstacles when adopting the optimization criteria  $P_a$  and  $P_l$ .

Robot	Without obstacles		With two obstacles	
	Optimization criteria			
	$P_a$	$P_l$	$P_a$	$P_l$
2R	1995	1572		
3R	10719	6716	16988	13807
4R	56548	25235	56646	46235

Table 2: Values of  $E_l$  for the experiments with zero and two obstacles when adopting the optimization criteria  $P_a$  and  $P_l$ .

Tables 1 and 2 show the values of  $E_a$  and  $E_l$  for the experiments with zero and two obstacles when adopting the optimization criteria  $P_a$  and  $P_l$ . In both cases, the *GA* optimization via the criterion  $P_l$  leads, in general, to smaller energy requirements. In particular, the reason for  $E_a$  being smaller with the  $P_l$  criterion is that the parameter tuning adopted in the experiments corresponds to a ‘weight’ ( $\alpha_{\phi} \cdot P_l = \text{weight} * \text{typical simulation value}$ ) that is lightly greater than the ‘weight’ of  $P_a$  in the fitness function. As a drawback, the trajectory reveals a higher ripple and presents a larger distance.

As expected, in a workspace with obstacles, the larger the number of dof the better the robot ability to maneuver and to reach the desired points. On the other hand, for actuator driving electronics with low efficiency (both in the  $P_a$  and  $P_l$  perspectives), the larger the number of dof the higher the total robot energy consumption.

## 7 Conclusions

An off-line *GA* trajectory planner for robots, based on the kinematics and the dynamics was presented. The algorithm is able to reach a determined goal with a reduced ripple both in the space trajectory and the time evolution. Moreover, any obstacles, in the workspace do not represent a difficulty for the algorithm to reach the solution. Since the *GA* uses the direct kinematics the singularities do not constitute a problem. Furthermore, the algorithm is easily generalized for redundant robots. In what concerns the dynamics, in the article were adopted two indices based on the actuator power consumption, namely the ‘absolute’ and the ‘dissipated’ power. In both perspectives, the power consumption is highly dependent on the number of dof and, therefore, to take advantage of the robot redundancy, it is required an efficient driving system.

## References

- A.B. Doyle, D.I Jones (1996), *Robot Path Planning with Genetic Algorithms*, 2<sup>nd</sup> Portuguese Conf. on Automatic Control, pp. 312-218, Porto, Portugal.
- A. Rana, A. Zalzalá (1996), *An Evolutionary Planner for Near Time-Optimal Collision-Free Motion of Multi-Arm Robotic Manipulators*, Int. Conf. Control, vol 1, pp 29-35
- David E. Goldberg (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison - Wesley.
- Filipe Silva, J. Tenreiro Machado (1999), *Energy Analysis During Biped Walking*, Proc. IEEE Int. Conf. Robotics and Automation Detroit, Michigan, pp. 59-64.
- Mingwu Chen, Ali M. S. Zalzalá (1997), *A Genetic Approach to Motion Planning of Redundant Mobile Manipulator Systems Considering Safety and Configuration*, J. Robotic Syst. vol. 14, n. 7, pp. 529-544.
- Naoyuki Kubota, Takemasa Arakawa, Toshio Fukuda (1997), *Trajectory Generation for Redundant Manipulator Using Virus Evolutionary Genetic Algorithm*, IEEE Int. Conf. on Robotics and Automation, pp. 205-210, Albuquerque, New Mexico.
- Q. Wang, A. M. S. Zalzalá (1996), *Genetic Control of Near Time-optimal Motion for an Industrial Robot Arm*, IEEE Int. Conf. on Robotics and Automation, pp. 2592-2597, Minneapolis, Minnesota.
- Thomas Bäck, Ulrich Hammel, Hans-Paul Schwefel (1997), *Evolutionary Computation: Comments on the History and Current State*, IEEE Trans. on Evolutionary Computation, Vol. 1, n. 1, pp. 3-17, April.
- Yaval Davidor (1991), *Genetic Algorithms and Robotics, a Heuristic Strategy for Optimization*, World Scientific.
- Z. Michalewicz (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag.
- E. J. Solteiro Pires, J. Tenreiro Machado (1999), *A Trajectory Planner for Manipulators Using Genetic Algorithms*, IEEE Int. Symp. on Assembly and Task Planning, pg. 163-168, Porto, Portugal, July.