Robot Trajectory Planning Using Multi-objective Genetic Algorithm Optimization

E.J. Solteiro Pires[1], J.A. Tenreiro Machado[2], and P.B. de Moura Oliveira[1]

[1] Universidade de Trás-os-Montes e Alto Douro, Dep. de Engenharia Electrotécnica,
Quinta de Prados, 5000–911 Vila Real, Portugal,
{epires,oliveira}@utad.pt, http://www.utad.pt/~epires
http://www.utad.pt/~oliveira

[2] Instituto Superior de Engenharia do Porto, Dep. de Engenharia Electrotécnica,
Rua Dr. António Bernadino de Almeida, 4200-072 Porto, Portugal
jtm@dee.isep.ipp.pt, http://www.dee.isep.ipp.pt/~jtm

Abstract. Generating manipulator trajectories considering multiple objectives and obstacle avoidance is a non trivial optimization problem. In this paper a multi-objective genetic algorithm is proposed to address this problem. Multiple criteria are optimized up to five simultaneous objectives. Simulations results are presented for robots with two and three degrees of freedom, considering two and five objectives optimization. A subsequent analysis of the solutions distribution along the converged non-dominated Pareto front is carried out, in terms of the achieved diversity.

1 Introduction

In the last twenty years genetic algorithms (GAs) have been applied in a plethora of fields such as: control, system identification, robotics, planning and scheduling, image processing, pattern recognition and speech recognition [1]. This paper addresses the planning of trajectories, meaning the development of an algorithm to find a continuous motion that takes the manipulator from a given starting configuration to a desired end position in the workspace without colliding with any obstacle.

Several single-objective methods for trajectory planning, collision avoidance and manipulator structure definition have been proposed. A possible approach consists in adopting the differential inverse kinematics, using the Jacobian matrix, for generating the manipulator trajectories [2,3]. However, this algorithm must take into account the kinematic singularities that may be hard to tackle. To avoid this problem, other algorithms for the trajectory generation are based on the direct kinematics [4,5,6,7,8].

Chen and Zalzala [2] propose a GA method to generate the position and the configuration of a mobile manipulator. In this report the inverse kinematics scheme is applied to optimize the least torque norm, the manipulability, the torque distribution and the obstacle avoidance. Davidor [3] also applies GAs to the trajectory generation by searching the inverse kinematics solutions to pre-defined end effector robot paths. Kubota et al. [4] study a hierarchical trajectory planning method for a redundant manipulator with a virus-evolutionary GA, running 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 Zalzala [5] develope 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. Chocron and Bidaud [9] propose an evolutionary algorithm to perform a task-based design of modular robotic systems. The system consists in a mobile base and an arm that may be built with serially assembled links and joints modules. The optimization design is evaluated with geometric and kinematic performance measures. Kim and Khosha [10] present the design of a manipulator that is best suited for a given task. The design consists of determining the trajectory and the length of a three degrees of freedom (*dof*) manipulator. Han et al. [11] describe a design method of a modular manipulator that uses the kinematic equations to determine the robot configuration and, in a second phase, adopts a GA to find the optimal length.

Gacôgne [12] presents a problem involving obstacle avoidance. He looks for an emergence of rules system for a mobile robot to have a good road-holding behavior in different playgrounds. A multi-objective genetic algorithm is used to find a short and readable solutions for every concrete problem.

Multi-objective techniques using GAs have been increasing in relevance as a research area. In 1989, Goldberg [13] suggested the use of a GA to solve multi-objective problems and since then other investigators have been developing new methods, such as multi-objective genetic algorithm (*MOGA*) [14], non-dominated sorted genetic algorithm (*NSGA*) [15] and niched Pareto genetic algorithm (*NPGA*) [16], among many other variants [17].

In this line of thought, this paper proposes the use of a multi-objective method to optimize a manipulator trajectory. This method is based on a GA adopting direct kinematics. The optimal manipulator front is the one that minimizes the objectives without any collision with the obstacles in the workspace. Following this introduction, the paper is organized as follows: section 2 formulates the problem and the GA-based method for its resolution. Section 3 presents several simulations results involving different robots, objectives and workspace settings. Finally, section 4 outlines the main conclusions.

2 Problem and Algorithm Formulation

This study considers robotic manipulators that are required to move from an initial point up to a given final configuration. Two and three *dof* planar manipulators (*i.e.* 2*R* and 3*R* robots) are used in the experiments with link lengths of one meter and rotational joints which are free to rotate 2π *rad*. To test a possible manipulator/obstacle collision, the arm structure is analyzed in order to verify if it is inside of any obstacle. The trajectory consists in a set of strings representing the joint positions between the initial and final robot configurations.

2.1 Representation

The path for a *iR* manipulator ($i = 2, 3$), at generation T, is directly encoded as vectors in the joint space to be used by the GA as:

$$[\{q_1^{(\Delta t,T)},..,q_i^{(\Delta t,T)}\},\{q_1^{(2\Delta t,T)},..,q_i^{(2\Delta t,T)}\},..,\{q_1^{((n-2)\Delta t,T)},..,q_i^{((n-2)\Delta t,T)}\}] \quad (1)$$

where i is the number of *dof* and Δt the sampling time between two consecutive configurations.

The joints values $q_l^{(j\Delta t,0)}$ $(j = 1, \dots, n-2;\ l = 1, \dots, i)$ are randomly initialized in the range $]-\pi, +\pi]$ *rad*. It should be noted that the initial and final configurations have not been encoded into the string because they remain unchanged throughout the trajectory search. Without losing generality, for simplicity, it is adopted a normalized time of $\Delta t = 0.1$ *sec*, because it is always possible to perform a time re-scaling.

2.2 Operators in the Multi-objective Genetic Algorithm

The initial population of strings is randomly generated. The search is then carried out among this population. Three different operators are used in the genetic planning: selection, crossover and mutation, as described in the sequel.

In what concerns the selection operator, the successive generations of new strings are reproduced on the basis of a Pareto ranking [13] with $\sigma_{\text{share}} = 0.01$ and $\alpha = 2$. To promote population diversity a metric count is used. This metric uses all solutions in the population independently of their rank to evaluate every fitness function. For the crossover operator it is used the simulated binary crossover (*SBX*)[15]. After crossover, the best solutions (among both parents and children) are chosen to form the next population. The mutation operator replaces one gene value with a given probability using the equation:

$$q_i^{(j\Delta t,T+1)} = q_i^{(j\Delta t,T)} + N(0, 1/\sqrt{2\pi}) \quad (2)$$

at generation T, where $N(\mu, \sigma)$ is the normal distribution function with average μ and standard deviation σ.

2.3 Evolution Criteria

Five indices $\{q, \dot{q}, p, \dot{p}, E_a\}$ (3) are used to qualify the evolving trajectory robotic manipulators. These criteria are minimized by the planner to find the optimal Pareto front. Before evaluating any solution all the values such that $|q_i^{((j+1)\Delta t,T)} - q_i^{(j\Delta t,T)}| > \pi$ are readjusted, adding or removing a multiple value of 2π, in the strings.

$$q = \sum_{j=1}^{n} \sum_{l=1}^{i} \left(\dot{q}_l^{(j\Delta t,T)}\right)^2 \quad (3a)$$

$$\dot{q} = \sum_{j=1}^{n} \sum_{l=1}^{i} \left(\ddot{q}_l^{(j\Delta t,T)}\right)^2 \quad (3b)$$

Table 1. Fronts parameters statistics

	Pareto front			Local front		
	κ	α	β	κ	α	β
Median	13.46	−8.32	−10.77	19.23	49.28	−13.02
Average	13.45	−7.40	−9.95	19.18	49.48	−13.19
Standard deviation	0.37	2.71	1.82	0.30	3.65	0.76

$$p = \sum_{j=2}^{n} d\left(p_j,\ p_{j-1}\right)^2 \tag{3c}$$

$$\dot{p} = \sum_{j=3}^{n} \left\{d\left(p_j,\ p_{j-1}\right) - d\left(p_{j-1},\ p_{j-2}\right)\right\}^2 \tag{3d}$$

$$E_a = (n-1)\Delta t\ P_a = \sum_{j=1}^{n}\sum_{l=1}^{i} \left|\tau_l . \Delta q_l^{(j\Delta t, T)}\right| \tag{3e}$$

The joint distance q (3a) is used to minimize the manipulator joints travelling distance. For a function $y = g(x)$ the curve length is defined by:

$$\int [1 + (\mathrm{d}g/\mathrm{d}x)^2]\mathrm{d}x \tag{4}$$

and, consequently, to minimize the curve length distance is adopted the simplified expression:

$$\int (\mathrm{d}g/\mathrm{d}x)^2 \mathrm{d}x = \int \dot{g}^2 \mathrm{d}x. \tag{5}$$

The joint velocity $\dot{q}$ is used to minimize the ripple in time evolution. The cartesian distance p (3c) minimizes the total arm trajectory length, from the initial point up to the final point, where p_j is the robot j intermediate arm cartesian position and $d(\cdot,\cdot)$ is a function that gives the distance between two arguments. The cartesian velocity is responsible for reducing the ripple in arm time evolution. Finally, the energy E_a in expression (3e), where τ_l are the robot joint torques, is computed assuming that power regeneration is not available by motors doing negative work, that is, by taking the absolute value of the power [18].

3 Simulation Results

In this section results of various experiments are presented. In this line of thought, subsections 3.1 and 3.2 present the trajectory optimization for the 2*R* and 3*R* robots respectively, for two objectives (2*D*). Finally, subsection 3.3 shows the results of a five dimensional (5*D*) optimization for a 2*R* robot.

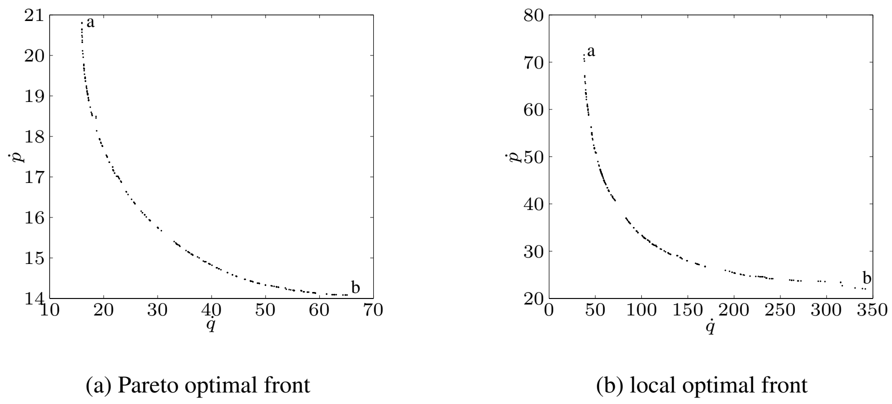


Fig. 1. Optimal fronts for the 2*R* robot

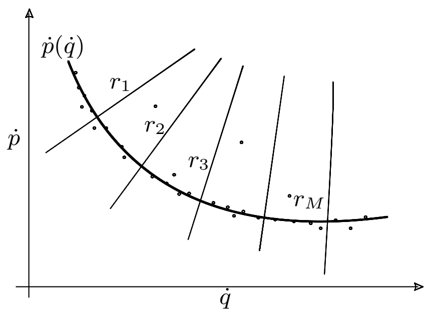


Fig. 2. Normal straight lines to the front obtained with $p(q)$ function

3.1 2*R* Robot Trajectory with 2*D* Optimization

The experiments consist on moving a 2*R* robotic arm from the starting configuration, defined by the joint coordinates $A \equiv \{-1.149, 1.808\}$ *rad*, up to the final configuration, defined by $B \equiv \{1.181, 1.466\}$ *rad*, in a workspace without obstacles. The objectives used in this section to optimize are the joint velocity $\dot{q}$ (3b) and the cartesian velocity $\dot{p}$ (3d).

The simulations results achieved by the algorithm, with $n = 9$ configurations, $T_t = 15000$ generations and $pop_{\text{size}=300}$, converge to two optimal fronts. One of the fronts (fig. 1(a)) corresponds to the movement of the manipulator around its base in the counterclockwise direction. The other front (fig. 1(b)) is obtained when the manipulator moves in the clockwise direction. The solutions *a* and *b*, shown in fig. 1 represent the best solution found for the $\dot{q}$ and $\dot{p}$ objectives, respectively.

In 66.6% of the 21 total number of runs, the Pareto optimal front was found. In all simulations for both cases the solutions converged to a front type which can be modelled by the following equation ($\kappa, \alpha, \beta \in \mathbb{R}$):

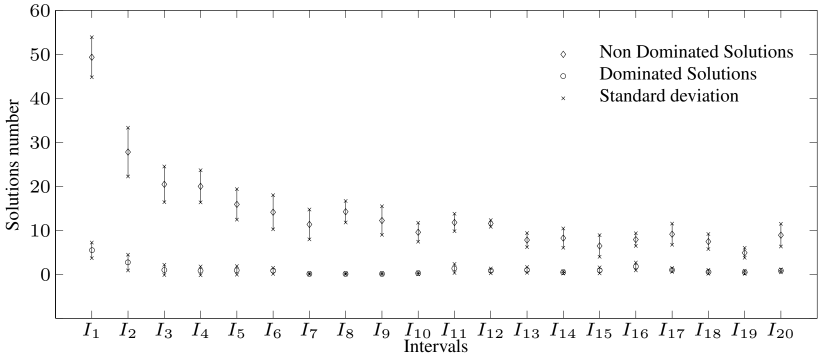
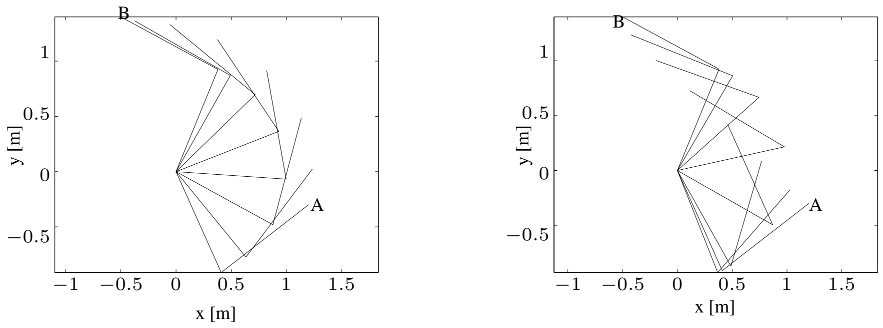


Fig. 3. Solution distribution statistics for the 2*R* robot



(a) $\dot{q}$ optimization (solution *a*)

(b) $\dot{p}$ optimization (solution *b*)

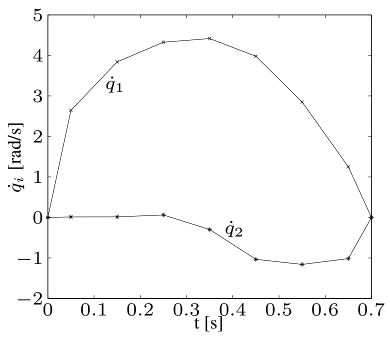
Fig. 4. Successive 2*R* robot configurations

Table 2. Range objectives in the 5*D* optimization

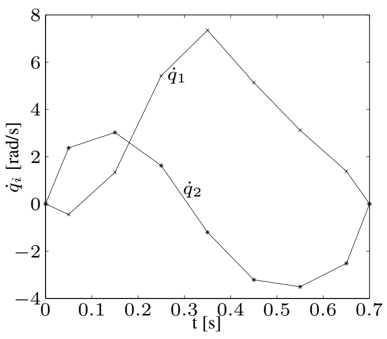
	q	$\dot{q}$	E_a	p	$\dot{p}$
min	79.8	18.2	1056.7	83.5	15.0
max	182.3	101.7	4602.7	121.8	56.4

$$\dot{p}(\dot{q}) = \kappa \frac{\dot{q} + \alpha}{\dot{q} + \beta} \tag{6}$$

The achieved median, average and standard deviation for the parameters κ, α and β of (6) are shown in table 1, both for the Pareto optimal and local fronts.



(a) $\dot{q}$ optimization (solution *a*)



(b) $\dot{p}$ optimization (solution *b*)

Fig. 5. Joint time evolution versus time for the 2*R* robot

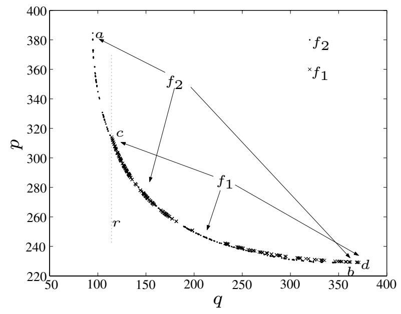


Fig. 6. Pareto optimal fronts, angular distance *vs*. cartesian distance optimization: $f_1 = \widehat{ab}$ – workspace without obstacles; $f_2 = \widehat{cd}$ – workspace with one obstacle

To study the solution front diversity, the approximated front was split into several intervals, limited by normal straight lines r_m (fig. 2), such that the front curve length is equal for all intervals. For any two consecutive normal straight lines an interval I_m ($m = 1, \dots, 19$) is associated, and the solutions located between these lines are counted. Figure 3 shows the solution distribution statistics achieved by all simulation runs. In this chart non-dominated and dominated solutions are represented, namely its average and its standard deviation. From this chart, it can be seen that the solutions are distributed by all intervals. However, the distribution is not uniform. This is due to the use of a sharing function in the attribute domain in spite of the objective domain. Moreover, the algorithm does not incorporate any mechanism to promote the development of well distributed solutions in the objective domain.

The results obtained for solutions *a* and *b*, of the Pareto optimal front in figure 1(a), are presented in figures 4 and 5. Comparing figures 4(a) and 5(a) with figures 4(b) and

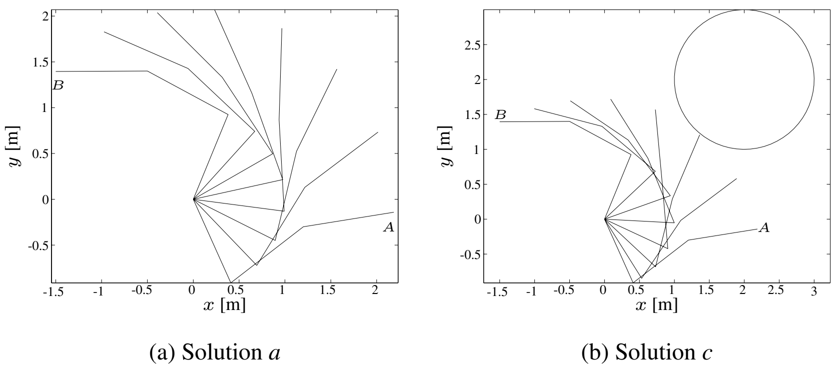


Fig. 7. Successive 3*R* robot configurations

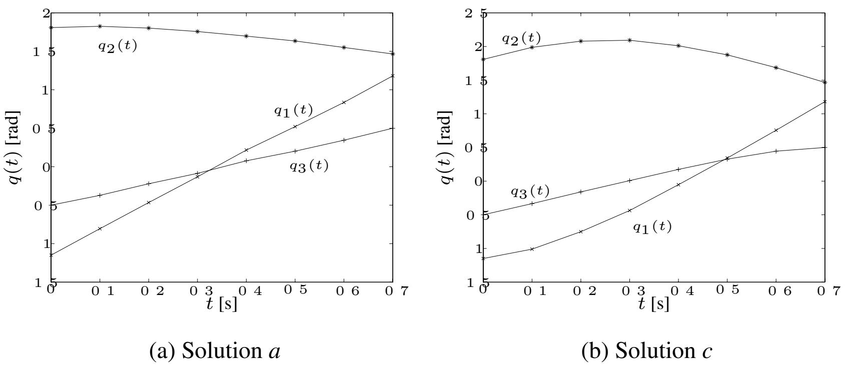
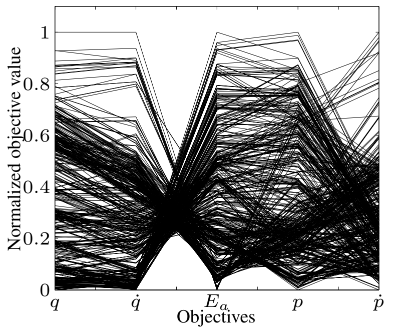


Fig. 8. Joint position of trajectory *vs.* time for the 3*R* robot

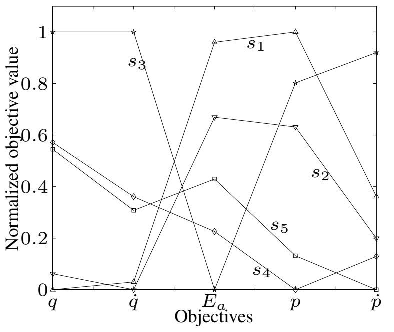
5(b) it is clear that the joint/cartesian time evolution for the optimal solutions a and b, respectively, is significantly different due to the objective considered. Between these extreme optimal solutions several others were found, that have a intermediate behavior, and which can be selected according with the importance of each objective.

3.2 3*R* Robot Trajectory with 2*D* Optimization

In this subsection a 3*R* robot trajectory is optimized using the objectives q (3a) and p (3c) in a workspace which may include a circle obstacle with center at $(x, y) = (2, 2)$ and radius $\rho = 1$. The initial and final configurations are $A \equiv \{-1.15, 1.81, -0.50\}$ *rad* and $B \equiv \{1.18, 1.47, 0.50\}$ *rad*, respectively. The T_t and pop_{size} parameters used are identical to those adopted in the previous subsection. The trajectories which collide with the obstacle are assigned a very high fitness value.



(a) Population tradeoffs



(b) Best solutions tradeoffs

Fig. 9. Normalized tradeoffs between q, $\dot{q}$, E_a, p and $\dot{p}$

For an optimization without any obstacle in the workspace the $f_2 = \widehat{ab}$ front (fig. 6) is obtained. However, when the obstacle is introduced the front is reduced to the $f_1 = \widehat{cd}$. Thus, only the objective q if affected by the introduction of the obstacle (figures 7 and 8). The solutions $\{a, b\}$ and $\{c, d\}$ represent the best solution found for the objectives $\{q, p\}$ for the 3*R* manipulator without and with obstacles, respectively.

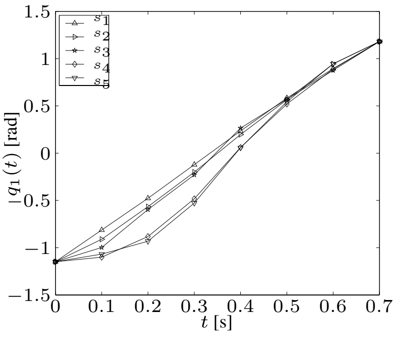
3.3 2*R* Robot Trajectory with 5*D* Optimization

Here, the 2*R* manipulator trajectory is optimized considering the five objectives described by equations (3). Figure 9 and 10 show the optimization results achieved with $T_t = 50000$ generations and $pop_{\text{size}} = 1000$.

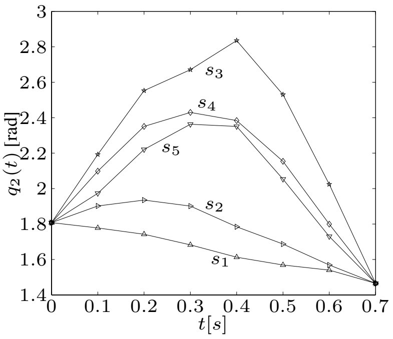
Table 2 contains the objectives range values archived in one simulation. Although, the 5*D* algorithms can't obtain so goods solutions as the 2*D* algorithm due to the significantly increase in the search complexity, the solutions have a good distribution (figure 9(a)) with values near to the ones for the 2*D* respective simulation. Figure 9 shows the normalized tradeoffs for the entire population and best solutions $s_i = \{s_1, s_2, s_3, s_4, s_5\}$ for each objective $O_i = \{q, \dot{q}, E_a, p, \dot{p}\}$. From figure 9(b) it can be concluded that q and $\dot{q}$ or p and $\dot{p}$ are conflicting objectives with a relative low tradeoff between them. On the other hand, the E_a objective presents the highest tradeoff among the others objectives. Figures 10 and 11 show the best solutions obtained for the objectives s_i. For the studied trajectory, the results indicate that as the manipulator moves near to its basis the energy consumed is lower (figures 11(a), 11(c) and 11(d)).

4 Summary and Conclusions

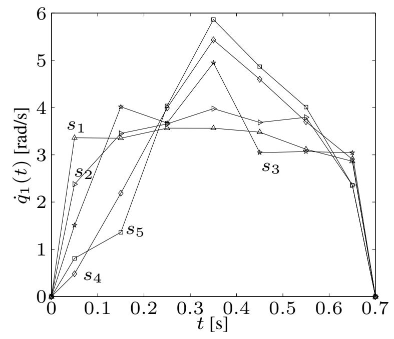
A multi-objective genetic algorithm robot trajectory planner, based on the kinematics approach, was proposed. The multi-objective genetic algorithm is able to reach optimal



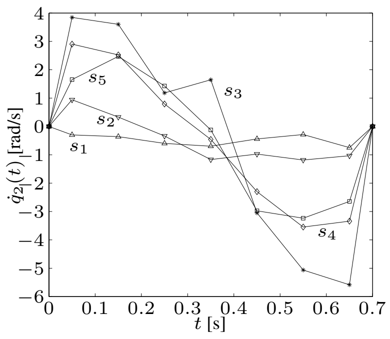
(a) Joint 1 position versus time



(b) Joint 2 position versus time



(c) Joint 1 time evolution versus time

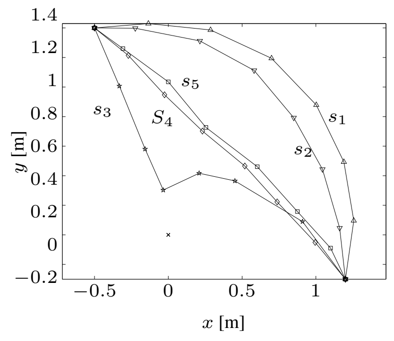


(d) Joint 2 time evolution versus time

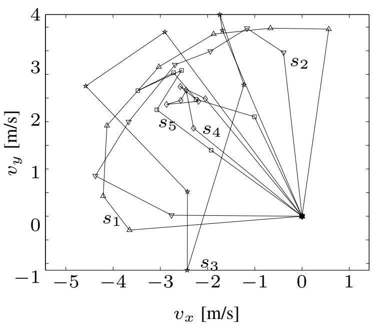
Fig. 10. Behavior of the best solutions

solutions regarding the optimization of multiple objectives. Simulation results were presented considering the optimization of two and five simultaneous objectives. The results obtained indicate that obstacles in the workspace may reduce the Pareto front length and for the case studied the single obstacle considered do not represent a difficulty for the algorithm to reach optimal solutions. Furthermore, the algorithm determines the non-dominated front maintaining a good distribution of solutions along the Pareto front.

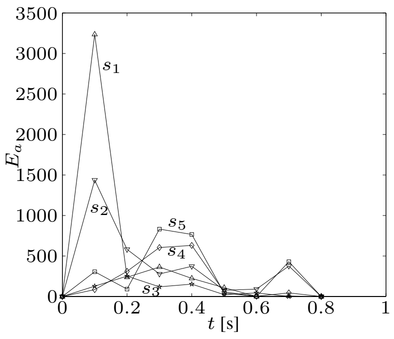
Acknowledgment. This paper is partially supported by the grant Prodep III (2/5.3/2001) from FSE.



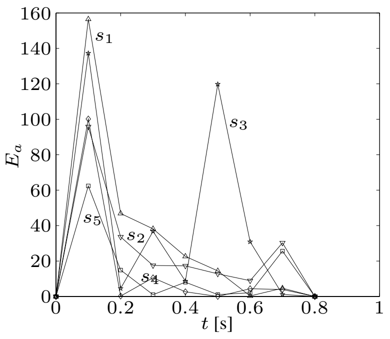
(a) Cartesian movement



(b) Cartesian time evolution



(c) Joint 1 energy required



(d) Joint 2 energy required

Fig. 11. Behavior of the best solutions (cont.)

References

1. Bäck, T., Hammel, U., Schwefel, H.P.: Evolutionary computation: Comments on the history and current state. IEEE Trans. on Evolutionary Computation **1** (1997) 3–17
2. Chen, M., Zalzala, A.M.S.: A genetic approach to motion planning of redundant mobile manipulator systems considering safety and configuration. Journal Robotic Systems **14** (1997) 529–544
3. Davidor, Y.: Genetic Algorithms and Robotics, a Heuristic Strategy for Optimization. Number 1 in Series in Robotics and Automated Systems. World Scientific Publishing Co. Pte Ltd (1991)
4. Kubota, N., Arakawa, T., Fukuda, T.: Trajectory generation for redundant manipulator using virus evolutionary genetic algorithm. In: IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico (1997) 205–210

5. Rana, A., Zalzala, A.: An evolutionary planner for near time-optimal collision-free motion of multi-arm robotic manipulators. In: UKACC International Conference on Control. Volume 1. (1996) 29–35
6. Wang, Q., Zalzala, A.M.S.: Genetic control of near time-optimal motion for an industrial robot arm. In: IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota (1996) 2592–2597
7. Pires, E.S., Machado, J.T.: Trajectory optimization for redundant robots using genetic algorithms. In Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.G., eds.: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000), Las Vegas, Nevada, USA, Morgan Kaufmann (2000) 967
8. Pires, E.J.S., Machado, J.A.T.: A GA perspective of the energy requirements for manipulators maneuvering in a workspace with obstacles. In: Proc. of the 2000 Congress on Evolutionary Computation, Piscataway, NJ, IEEE Service Center (2000) 1110–1116
9. Chocron, O., Bidaud, P.: Evolutionary algorithms in kinematic design of robotic system. In: IEEE/RSJ International Conference on Intelligent Robotics and Systems, Grenoble, France (1997) 279–286
10. Kim, J.O., Khosla, P.K.: A multi-population genetic algorithm and its application to design of manipulators. In: IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems, Raleight, North Caroline (1992) 279–286
11. Han, J., Chung, W.K., Youm, Y., Kim, S.H.: Task based design of modular robotic manipulator using efficient genetic algorithm. In: IEEE Int. Conf. on Robotics and Automation, Albuquerque, New Mexico (1997) 507–512
12. Gacôgne, L.: Multiple objective optimization of fuzzy rules for obstacles avoiding by an evolution algorithm with adaptative operators. In: In Proceedings of the Fifth International Mendel Conference on Soft Computing (Mendel'99), Brno, Czech Republic (1999) 236–242
13. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison – Wesley (1989)
14. Fonseca, C.M., Fleming, P.J.: An overview of evolutionary algorithms in multi-objective optimization. Evolutionary Computation Journal **3** (1995) 1–16
15. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley-Interscience Series in Systems and Optimization. (2001)
16. Horn, J., Nafploitis, N., Goldberg, D.: A niched pareto genetic algorithm for multi-objective optimization, Proceedings of the First IEEE Conference on Evolutionary Computation (1994) 82–87
17. Coello, C., Carlos, A.: A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowledge and Information Systems **1** (1999) 269–308
18. Silva, F., Tenreiro Machado, J.: Energy analysis during biped walking. In: Proc. IEEE Int. Conf. Robotics and Automation, Detroit, Michigan (1999) 59–64