European Journal of Mechanics A/Solids 42 (2013) 210–218

Contents lists available at SciVerse ScienceDirect

European Journal of Mechanics A/Solids

journal homepage: www.elsevier.com/locate/ejmsol

Evolutionary indirect approach to solving trajectory planning problem for industrial robots operating in workspaces with obstacles

Fares J. Abu-Dakka*, Francisco Rubio, Francisco Valero, Vicente Mata

Centro de Investigación de Tecnología de Vehículos CITV, Universidad Politécnica de Valencia, Valencia – Edificio 5E – Camino de Vera s/n, 46022 Valencia, Spain

ARTICLE INFO

Article history:
Received 1 July 2012
Accepted 30 May 2013
Available online 19 June 2013

Keywords:
Trajectory planning
Obstacle avoidance
Genetic algorithms

ABSTRACT

In this paper, an indirect method for trajectory planning for industrial robots has been addressed using an evolutionary algorithm. The algorithm is divided into three stages: (1) The acquisition of Adjacent Configurations (AC) for Path Planning subjected to kinematics, geometric and obstacle avoidance constraints. (2) The acquisition of a collision-free path between initial and goal robot configurations. This path consists of a set of ACs, and (3) The acquisition of a temporal history of the evolution for the robot joint coordinates, by minimizing the required time subjected to actuator limits. This algorithm has been evaluated by comparing the results with the direct procedures proposed by Rubio articles in 2009 and 2010.

1. Introduction

In the last few decades, the prevalence of robots has grown in many areas. In addition to industrial applications, robots are also used in surgery, agriculture, underwater, and for transportation. In industrial applications, they have many purposes like; pick and place operations, assembly tasks, spray-painting, and many other tasks.

In general, the purpose of robotic systems is to perform such tasks smoothly in as little time as possible. Thus, a procedure of three stages is proposed to obtain an offline minimum time trajectory planning. The offline trajectory planning is justified as a large number of robotic applications work in a repetitive manner.

Path planning and trajectory planning problems are two distinct parts of the robotics that are intimately related. Actually, a clear difference exists between those algorithms devoted to path planning problem and those devoted to determining the optimal trajectory for robotic systems. The first ones try, essentially, to obtain a sequence ("a path") of robot configurations between an initial and goal configurations that fulfills some conditions, mainly, collision avoidance. Whereas, the second one try to obtain a temporal history of the evolution for the robot joint coordinates by minimizing aspects such as; the required time or the energy consumption. Therefore, path planning is a subset of trajectory planning, wherein the dynamics of the robot are neglected. In indirect methods, the path is searched first, and then finding a time-optimal time scaling for the path subject to the dynamic constraints of the manipulator; such approaches are known as decoupled (indirect) approaches (Piazzi and Visioli, 1997, 2000; Saramago and Steffen, 2001; Plessis and Snyman, 2003; Behzadipour and Khajepour, 2006; Valero et al., 2006; Bertolazzi et al., 2007; Gasparetto and Zanotto, 2007; Saravanan et al., 2009). Otherwise, in the direct approaches of trajectory planning, the search takes place in the system's state space (Rubio et al., 2009, 2010; Abdel-Malek et al., 2006). Most of the existing methods belong to one of these types, although the indirect methods are the most widely used.

A strategy to optimize the end-effector trajectory and dynamic behavior in a restricted workspace through spline interpolation is presented by Saramago and Steffen (1998). Lin et al. (1983) introduced the first formulation of the problem of finding the optimal curve interpolating a sequence of nodes in the joint space, subject to kinematic constraints. The same formulation then solved by means of cubic B-splines (Wang and Horng, 1990). The interpolation using cubic polynomial functions is used in multi-objective optimization problem, aim to minimize the time of motion and the mechanical energy of the actuators (Saramago and Steffen, 2001). Trigonometric splines are used to ensure the continuity of the jerk (Piazzi and Visioli, 2000). Moreover, different harmonic functions are used for interpolation (Rubio et al., 2009, 2010).

* Corresponding author. Tel.: +34 96 387 70 07; fax: +34 96 387 76 29.
E-mail address: fares.abudakka@gmail.com (F.J. Abu-Dakka).

0997-7538/$ – see front matter © 2013 Elsevier Masson SAS. All rights reserved.
http://dx.doi.org/10.1016/j.euromechsol.2013.05.007

In the literature, readers can find many methods to solve the trajectory planning problem, such as: the Sequential Unconstrained Minimization Technique (SUMT) by Saramago and Steffen (1998, 1999, 2000, 2001) and Saramago and Ceccareli (2002), the Interval Analysis by Aurelio and Visioli (2000), the Sequential Quadratic Programming (SQP) by Gasparetto and Zanotto (2007) and Chettibi et al. (2004), and the Numerical Iterative Procedure by Elnagar and Hussein (2000). The Traditional derivative-based optimization methods are designed to solve continuous and differentiable minimization problems, as they use derivatives to determine the direction of descent. However, using derivatives is often ineffective with discontinuous, non-differentiable or stochastic objective functions. For non-smooth problems, methods such as the genetic algorithm (GA) are effective alternatives.

In the last two decades, evolutionary algorithms such as Niched Pareto Genetic Algorithm (Horn et al., 1994), Multi-Objective Genetic Algorithms (Fonseca and Fleming, 1995), Elitist Non-dominated Sorting Genetic Algorithm (Deb et al., 2002), Multi-Objective Differential Evolution (Babu and Anbarasu, 2005), and Steady-State Genetic Algorithms (SSGA) (Abu-Dakka et al., 2007a, 2008) among many others, have been applied in the fields of robotics, planning, control, system identification, etc.

In this paper, a composite of three evolutionary algorithms are used to solve the trajectory planning problem. (1) A SSGA is used to optimize the path between adjacent configurations, subjected to collision avoidance constraints. (2) A Parallel Genetic Algorithm (PGA) procedure is used to obtain a collision-free path (sequence of adjacent configurations). (3) A PGA approach aims to schedule the time intervals between each adjacent configurations along the path, such that the total traveling time is minimized using Clamped Cubic Spline (CCS). This approach is subjected to kinematics and dynamics constraints. A comparison of results is presented between the proposed indirect method and the direct methods proposed by Rubio et al. (2009, 2010). Rubio et al. (2009) proposed a simultaneous direct approach for the trajectory planning problem for industrial robots in environments with obstacles, where the trajectory was created gradually as the robot moves. Their method deals with the uncertainties associated with lack of knowledge of kinematic properties of via points since they are generated as the algorithm evolves. One year later, the same authors Rubio et al. (2010) tested the simultaneous approach with different interpolation functions.

2. Robot & workspace modeling

In this paper, a wired model for robotic systems has been considered (Valero et al., 1996), Fig. 1. The robot configuration is defined in joint coordinates $C^j(q_i)$, while the workspace is defined in Cartesian coordinates. Moreover, the robot configuration can be expressed in Cartesian coordinates through a set of points called significant points $\gamma_m^j(q_i)$ and interesting points $\lambda_m^j(q_i)$, see Fig. 1, to facilitate the collision avoidance process. The selection of significant points is made based on the degrees of freedom of the robot in the way that they should be minimum as much as possible to define without ambiguity the configuration of the robot. It is important to emphasize that they do not constitute an independent set of coordinates. In the other hand, the interesting points are obtained from the significant points and the geometric characteristics of the robot. Thus, the robot configuration can be expressed in Cartesian coordinates through those points $C(\gamma_m^j,\ \lambda_m^j)$. It should be noted that any robotic system can be modeled in this way by just selecting and choosing appropriately those significant points that best describes the robotic system.

An application example, 4 significant points $\gamma_m^j = (\gamma_1^j,\ \gamma_2^j,\ \gamma_3^j,\ \gamma_4^j)$ and 4 interesting points $\lambda_m^j = (\lambda_1^j,\ \lambda_2^j,\ \lambda_3^j,\ \lambda_4^j)$, Fig. 1, are used to describe the geometry of PUMA 560 robotic system. By comparison, authors Rubio et al. (2009, 2010) used 7 significant points and 5 interesting points.

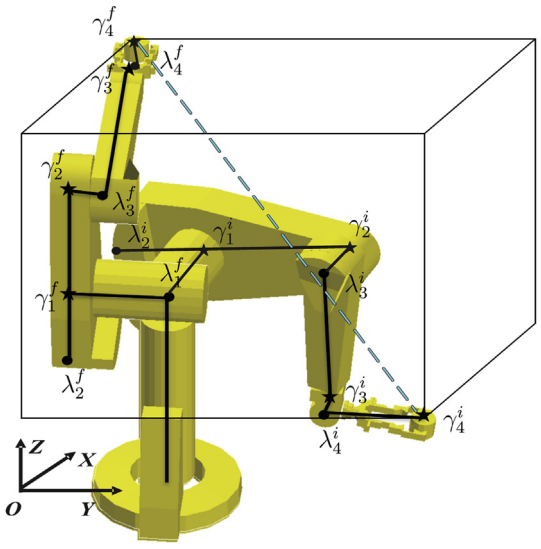


Fig. 1. Robot & workspace model.

In this paper, static obstacles are considered. To facilitate and systematize the calculation of the distances between the robot links and the obstacles, generic obstacle models have been constructed in terms of a combination of three basic patterns: spheres, cylispheres, and quadrilateral planes since they are computationally simple.

The workspace has been modeled in a rectangular prism between the initial $C^{\text{init}}(q_i)$ and goal $C^f(q_i)$ robot configurations. The end points of the prism's diagonal (represented by γ_4^i and γ_4^f in Fig. 1) are corresponding to the positions in Cartesian coordinates of the end-effector of the $C^{\text{init}}(q_i)$ and $C^f(q_i)$ respectively. The prism edges are parallel to the Cartesian reference system.

A uniform grid of points is considered inside the prism. These points are far a magnitude small enough (Δx, Δy, Δz) to prevent the existence of obstacles between adjacent points in the grid. Thus, the workspace contains a discrete set of configurations such that the position of the end-effector for each configuration must belong to the previously defined grid.

3. Adjacent configurations optimization

The basic stage of this approach is the generation of a discrete space of configurations based on the definition of adjacent configurations. The acquisition of a feasible configuration C^{j+1} that is adjacent to a given one C^j that satisfies the following three conditions (Abu-Dakka, 2011; Abu-Dakka et al., 2007b):

1. The distance between the end-effector position of C^{j+1} and C^j is smaller than the size of the smallest obstacle in the workspace.
   In case of PUMA 560, $\left\|\gamma_4^{j+1} - \gamma_4^j\right\|$ < the size of the smallest obstacle.
2. No obstacle can be placed between the configurations C^{j+1} and C^j. This is to verify that the distances of each pair of significant points between configurations C^{j+1} and C^j are smaller than the smallest obstacle in the workspace.
   In case of PUMA 560, $\left\|\gamma_l^{j+1} - \gamma_l^j\right\|$ < the size of the smallest obstacle. l = 1,2,3.

3. The following expression is minimized:

$$\left\|C^{j+1}-C^{j}\right\| = A\cdot\sum_{l=1}^{m}\left(\left(\gamma_l^{j+1}-\gamma_l^{j}\right)_x^2+\left(\gamma_l^{j+1}-\gamma_l^{j}\right)_y^2+\left(\gamma_l^{j+1}-\gamma_l^{j}\right)_z^2\right)+B\cdot\sum_{i=1}^{\mathrm{dof}}\left(q_i^{f}-q_i^{j+1}\right)^2 \quad (1)$$

where: dof = Robot Degree of Freedom. $m = 4$ and dof = 6 for PUMA 560. A and B are coefficients to adjust the results. The first component of the objective function is the distance between significant points of the two adjacent configurations to make them close. The second component is the distance between the current configuration C^{j+1} and the final configuration C^{f} in joint coordinates to make it converge. The Rubio et al. (2009, 2010)'s algorithms by comparison, tried to minimize the distance between significant points.

3.1. SSGA for adjacent configurations

In this stage, the generation of collision-free and discrete configurations will take place. A methodology of two distinct routines has been constructed to obtain a collision-free robot configuration C^{j+1} adjacent to C^{j}. In the first place, the inverse kinematic problem (Craig, 2005) will be used to find the C^{j+1} for a given end-effector position γ_4. If the new configuration C^{j+1} doesn't fulfill the conditions, a SSGA procedure will be used to solve the problem. The SSGA uses overlapping populations. This means, the ability to specify how much of the population should be replaced in each generation. Newly generated offspring are added to the population, and then the worst individuals are destroyed. Each chromosome consists of i genes and represents a robot configuration.

$$\text{chromosome} = \{q_1,\ q_2,\ \cdots,\ q_i\} \quad (2)$$

where: $i = 6$ for the PUMA 560 robot.

The population in this algorithm consists of 30 individuals (chromosomes). The initial values for each gene in the initial population have been selected randomly from the interval $[q_{i,\ \min},\ q_{i,\ \max}]$ (the maximum and minimum values of the joint variables of the robot). However, by looking to the adjacent configuration definition, it can be seen that the displacement from the configuration j to $j+1$ is very small. Consequently, that interval can be modified to a new one by adding/subtracting an arbitrary small amount Δq to q_i^j. This Δq depends on the discrete increment size. Therefore, the new interval for each q_i can be calculated as shown in the flow chart in Fig. 2.

A roulette-wheel selection method is applied to select individuals for crossover and mutation. This method based on the magnitude of the fitness score of an individual relative to the rest of the population. The higher score, the more likely an individual will be selected.

4. Path planning

In path planning problems, the number of feasible paths between the initial and final positions of a robot are often very large, and the goal is not necessarily to determine the best solution, but to obtain an acceptable one according to certain requirements and constraints. Various search methods have been developed (e.g., calculus-based methods, enumerative schemes, random search algorithms, etc.) for the robot path-planning problem. In this paper, a GA has been used to minimize the traveling distance of the significant points between the initial and final configurations avoiding obstacles.

The GA for path planning uses parallel populations with a migration technique. The GA has multiple, independent populations. Each population evolves using SSGA, but at each generation, some individuals migrate from one population to another. The migration algorithm is deterministic stepping-stone; each population migrates a fixed number of its best individuals to its neighbor. The master population is updated each generation with the best individual from each population.

The chromosome consists of a set of genes, each gene represents the end-effector position. The gene consists of three cells that represent the x, y, and z coordinates of the end-effector position.

$$\text{chromosome} = \left\{(X_1,\ Y_1,\ Z_1),\ \cdots,\left(X_f,\ Y_f,\ Z_f\right)\right\} \quad (3)$$

The objective here is to minimize the summation of the distances between the significant points between each pair of adjacent configurations along the path (Abu-Dakka, 2011):

$$\text{Minimize}\left\{\sum_{i=1}^{n-1}\sum_{j=1}^{m}\sqrt{\left(\gamma_j^{i+1}-\gamma_j^{i}\right)_x^2+\left(\gamma_j^{i+1}-\gamma_j^{i}\right)_y^2+\left(\gamma_j^{i+1}-\gamma_j^{i}\right)_z^2}\right\} \quad (4)$$

5. Trajectory planning

In this section, an optimization algorithm has been built using parallel-populations genetic algorithms to obtain minimum time cubic spline trajectories for a given sequence of configurations subjected to dynamic constraints of the robot. This choice can be justified by the well-known characteristics of cubic spline functions; continuity of second order is guarantied and their lower order greatly limits oscillations. The velocities at the ends of the path are considered to be *Zero*, such that $v_0 = 0$ and $v_n = 0$.

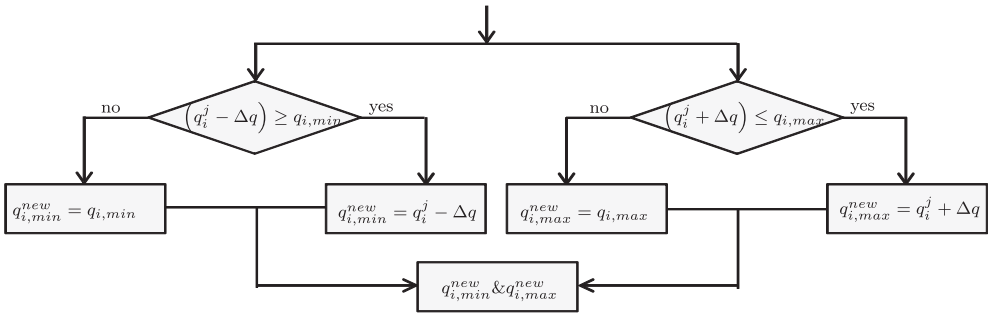


Fig. 2. Joints interval determination.

5.1. Objective function

$$\text{Minimize } \sum_{j=1}^{n-1} t_j \tag{5}$$

subjected to

$$\text{Kinematic constraints}\begin{cases}\text{Joint velocities}: & \left|\dot{q}_i^j(t)\right| \le \dot{q}_i^{\max}\\ \text{Joint accelerations}: & \left|\ddot{q}_i^j(t)\right| \le \ddot{q}_i^{\max}\\ \text{Joint jerks}: & \left|\dddot{q}_i^j(t)\right| \le \dddot{q}_i^{\max}\end{cases} \tag{6}$$

$$\text{Dynamic constraints}\begin{cases}\text{Joint torques}: & \left|\tau_i^j(t)\right| \le \tau_i^{j,\max}\\ \text{Joint power}: & \left|P_i^j(t)\right| \le P_i^{j,\max}\\ \text{Joint energy}: & \left|E_i^j(t)\right| \le E_i^{j,\max}\end{cases} \tag{7}$$

$$\text{Payload constraint} \quad Fg_{\min} \le F \le Fg_{\max} \tag{8}$$

where: $i = 1$ to the number of the degrees of freedom of the robot, $j = 1$ to the no. of nodes in the trajectory, F is the grasping force, $Fg_{\min} = 0$ and $Fg_{\max} = 60$ N are the minimum and maximum grasping forces used in the industrial application example, Section 6.3. The grasped object masses (payload) used are 1 kg and 0.5 kg in the applications 1 and 2 respectively.

For industrial applications, the speed of operation affects the productivity. To maximize the speed of operation, the traveling time for the robot should be minimized. Thus, the optimization problem is to adjust the time intervals between each pair of adjacent configurations such that the total traveling time is minimal. Specifically, the problem is to determine a set of optimum values for time intervals $t_1, t_2, \ldots, t_{n-1}$. Note that there are N joints which must be considered simultaneously.

Let Q be the set of m configurations that compose the path, $Q_j(q_{i,j})$ and $Q_{j+1}(q_{i,j+1})$ two adjacent configurations belonging to Q which are expressed in joint variables. The trajectory between these two configurations can be written in a polynomial form as,

$$q_{ij} = a_{ij} + b_{ij}t + c_{ij}t^2 + d_{ij}t^3 \tag{9}$$

where: $i = 1, 2, \ldots$, dof and $j = 1, 2, \ldots, m-1$.

The smoothness of the trajectory can be guaranteed by imposing the following conditions:

- Position;

For each interval j the initial and final position must fit Q_j and Q_{j+1}, obtaining $(2\cdot\text{dof}\cdot(m-1))$ equations.

- Velocity;

The velocities at the ends of the path must be zeros. This will contribute $(2\cdot\text{dof})$ equations. On the other hand, between intermediate configurations, the velocities at the end of an interval must be the same as the initial velocities of the following interval. This will contribute $(\text{dof}\cdot(m-2))$ equations.

$$\dot{q}_{ij}\left(t = \sum_{j=1}^{n} t_j\right) = \dot{q}_{i,j+1}(0) \tag{10}$$

- Acceleration;

In the intermediate configurations, the acceleration at the end of an interval must be the same as the one at the beginning of the next interval. This will contribute $(\text{dof}\cdot(m-2))$ equations.

$$\ddot{q}_{ij}\left(t = \sum_{j=1}^{n} t_j\right) = \ddot{q}_{i,j+1}(0) \tag{11}$$

The previous conditions define a $(4\cdot\text{dof}\cdot(m-1))$ linear independent system of equations when the time between successive configurations of the sequence Q is known.

A GA procedure with parallel populations with migration technique has been implemented to optimize the time intervals needed to move the robot between adjacent configurations in the pursued trajectory.

6. Numerical examples

Object Oriented C++ code has been implemented and executed using a computer with Intel Xeon CPU E5440 @ 2.83 GHz and 8 GB of RAM. For genetic algorithm procedures, the MIT Genetic Algorithm Library (Wall, 1996) has been used and adapted to the problem environment.

In this Section, application examples are particularized for the PUMA 560 robotic system and have been implemented and analyzed to evaluate the efficiency of the proposed algorithms. Three operational parameters have been used for the evaluation:

(a) Execution Time (t_e): The time needed for the robot to execute the trajectory.
(b) Traveled Distance (d_s): The summation of the distances between significant points of each pair of adjacent configurations along the path.
(c) Computational Time (t_c): The time consumed by the computer for calculations.

6.1. First Group: Comparison with Rubio et al. (2009)

This group consists of 20 examples obtained on the basis of 5 different initial and final robot configurations. Each one has been solved in 4 different workspaces. starting from the case without obstacles finishing with the case of 3 obstacles. The authors Rubio et al. (2009) compared their results by using three different approaches; A*, uniform cost (UC), and greedy (G).

The Cost function $H(j+1)$ associated to UC strategy allows the robot to arrive the node $j+1$ from node j, and can be separated into two terms:

(a) $f(j)$ is an indicator of the cost to reach node j. This value can be calculated since the configuration C^j has been reached.
(b) $c(j, j+1)$ is an indicator of the cost to go from configuration C^j to another adjacent C^{j+1}.

The total cost function associated to the UC can be grouped like this:

$$H(j+1) = f(j) + c(j, j+1) \tag{12}$$

The A* algorithm is based in heuristic functions. The A* algorithm in Rubio et al. (2009) had been created by adding an estimation to the UC function to estimate the cost to go from the

current configuration of the robot to the final one C^f. Thus, the terms of the A* are the previous terms of the UC plus the following one:

(c) $h(j+1,f)$ that corresponds with an estimation of the cost to go from adjacent configuration C^{j+1} to final configuration C^f. It is a heuristic function, in so far as an under estimation is produced about what could cost to arrive from the current node to the final one.

The total cost function associated to the A* algorithm can be grouped like this:

$$H(j+1) = f(j) + c(j,j+1) + h(j+1,f) \tag{13}$$

The cost function associated to a greedy search only considers the estimation of the cost to go from the current configuration to the goal (final) configuration. The total cost function associated to the greedy search can be expressed as this:

$$H(j+1) = h(j+1,f) \tag{14}$$

Rubio et al. (2009) stated that, the A* algorithm and the greedy strategy are different with different properties. The A* algorithm is complete, and it gives optimal results (sure, in terms of the cost function used), while the greedy strategy, does not guarantee that the solution is optimal and even may not find any solution.

Figs. 3–7 summarize the main comparisons of this group of examples. For more details about the Rubio procedure, please refer to Rubio et al. (2009).

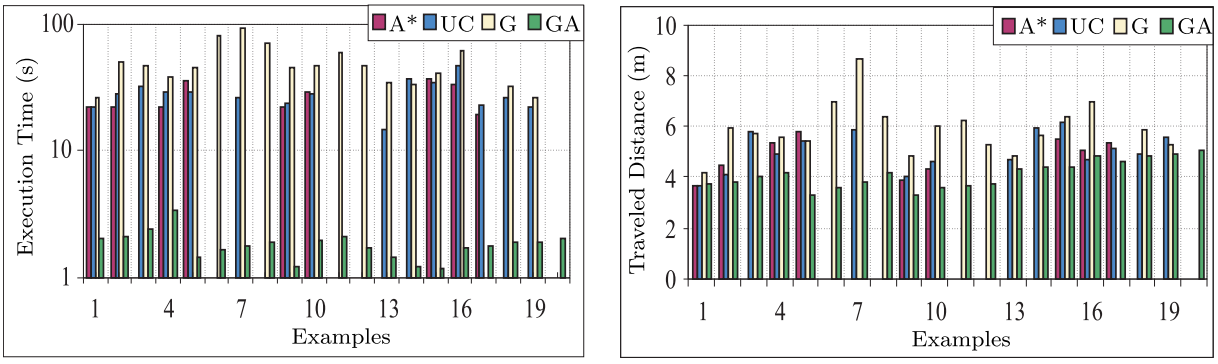
6.2. Second Group: Comparison with Rubio et al. (2010)

This group consists of 20 examples obtained on the basis of 10 different initial and final robot configurations. Each one has been solved in 2 different workspaces; the case without obstacles and the case with 1 obstacle. The authors Rubio et al. (2010) used harmonic functions for interpolation and they analyzed their results by using three different interpolation cases; F5, F7, and F8.

The F5, F7 and F8 functions represent mainly harmonic interpolation functions to model the joint kinematic evolution though the complete trajectory and they are used because they prevent the dynamic incompatibilities from appearing. For example, F5 uses the following equation:

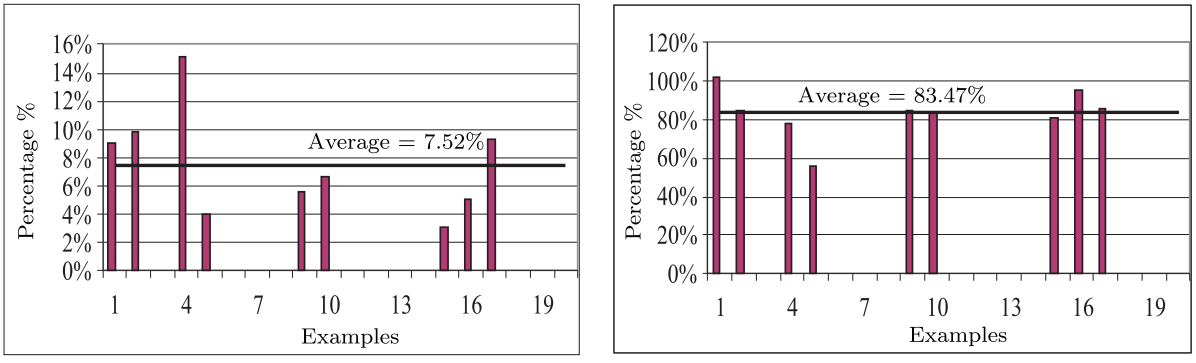
$$q_{ij} = a_{ij}\sin(t) - b_{ij}\cos(2t) + c_{ij}\sin(3t) - d_{ij}\cos(4t) \tag{15}$$

In, the other hand, F7 uses the following equation.



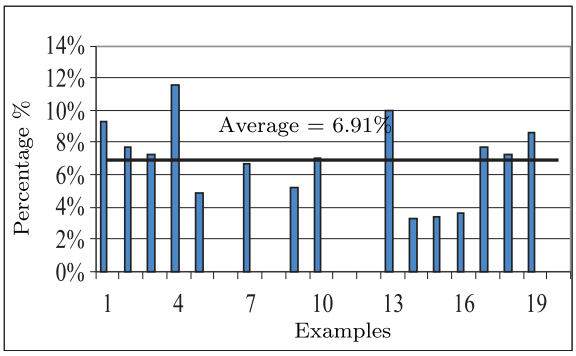
(a) Execution Time (b) Traveled Distance

Fig. 3. First Group, Comparison with Rubio et al. (2009), Comparing Execution Time and Traveled Distance.

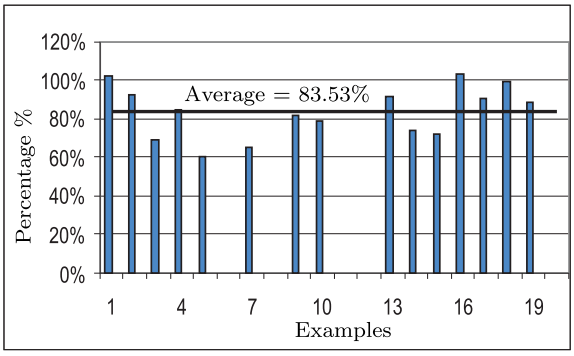


(a) GA execution time with respect to A* in (Rubio et al., 2009) (b) GA distance with respect to A* in (Rubio et al., 2009)

Fig. 4. First Group, Average of GA results with respect to A* in Rubio et al. (2009).

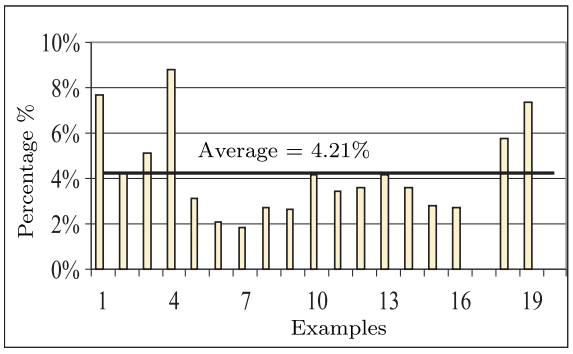


(a) GA execution time with respect to UC in (Rubio et al., 2009)

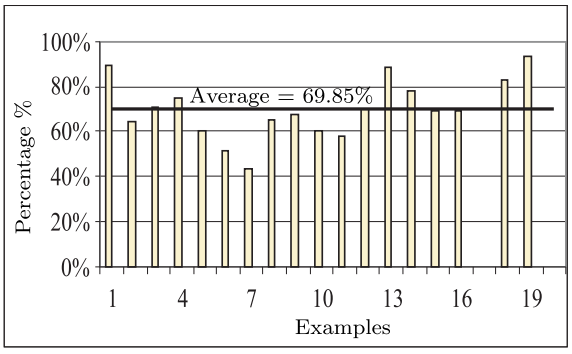


(b) GA distance with respect to UC in (Rubio et al., 2009)

Fig. 5. First Group, Average of GA results with respect to UC in Rubio et al. (2009).



(a) GA execution time with respect to G in (Rubio et al., 2009)



(b) GA distance with respect to G in (Rubio et al., 2009)

Fig. 6. First Group, Average of GA results with respect to G in Rubio et al. (2009).

$$q_{ij} = \cos(t)(\sin(t)(a_{ij}\sin(t) + b_{ij}) + c_{ij}) + d_{ij} \quad (16)$$

while the F8 uses

$$q_{ij} = \sin(t)(\cos(t)(a_{ij}\sin(t) + b_{ij}) + c_{ij}) + d_{ij} \quad (17)$$

In their paper, they tried to analyze which one gives the best results.

The next Table 1 and Figs. 8–10 show the comparisons of this group of examples. For more details about the Rubio procedure, the reader can refer to Rubio et al. (2010).

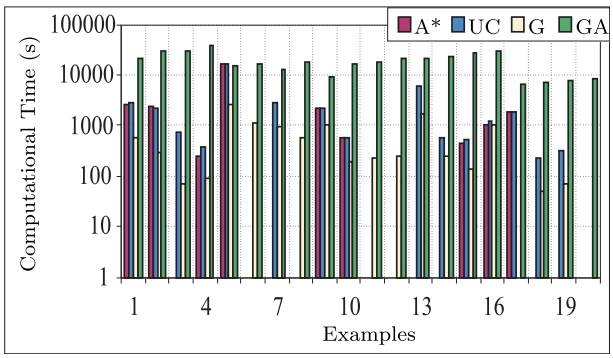


Fig. 7. First Group, Computational Time (s) Comparison with Rubio et al. (2009).

6.3. Industrial application

This group consists of scenarios of industrial applications to demonstrate the ability of the algorithm to adapt to any environment. The illustrative manipulator task consists in transporting a payload from an initial configuration to a final one.

6.3.1. Application # 1

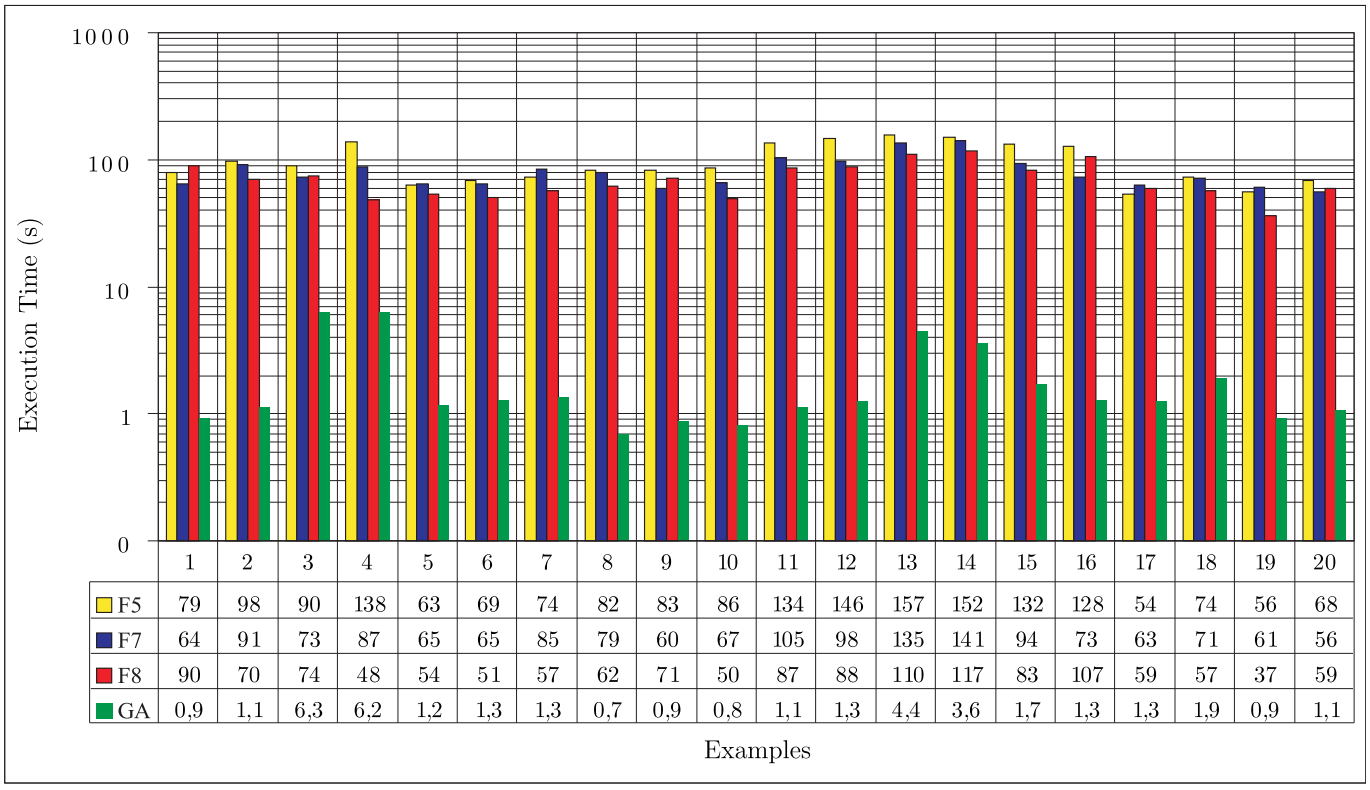
In the example shown in Fig. 11, the execution time $t_e = 1.52102$ s, the traveled distance $d_s = 3.2619$ m, and the computational time $t_c = 11893$ s.

6.3.2. Application # 2

In the example shown in Fig. 12, the execution time without and with obstacles $t_e = 2.42217$ and 3.85854 s respectively, the traveled distance without and with obstacles $d_s = 4.7870$ and 5.2227 m respectively, and the computational time without and with obstacles $t_c = 12915$ and 57080 s respectively.

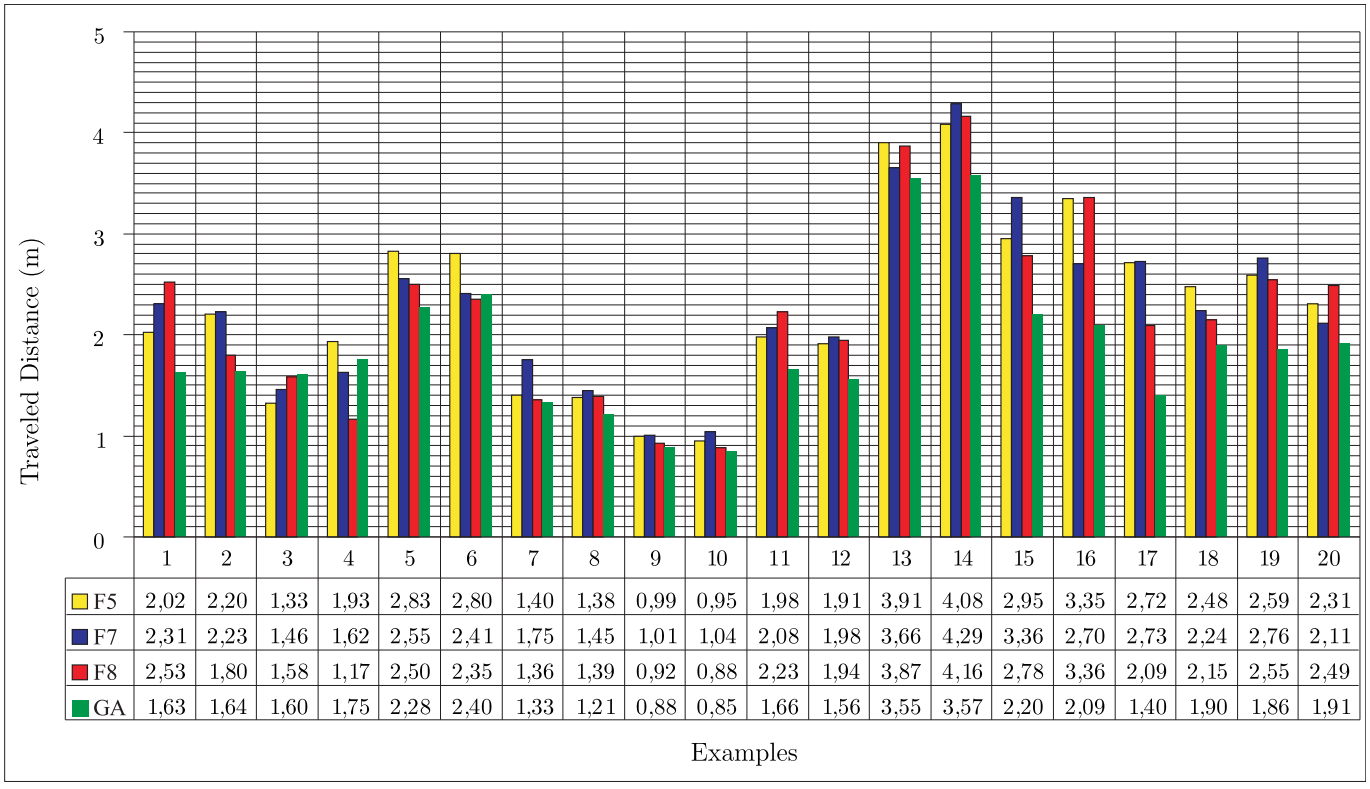
Table 1
Average Values of t_e = Execution Time, d_s = Traveled Distance, and t_c = Computational Time.

	$t_e(s)$	$d_s(m)$	$t_c(s)$
GA	1.9664	1.8639	4411
F5 in Rubio et al. (2010)	98.1163	2.3053	5429
F7 in Rubio et al. (2010)	81.6604	2.2872	2329
F8 in Rubio et al. (2010)	71.5478	2.2044	2854



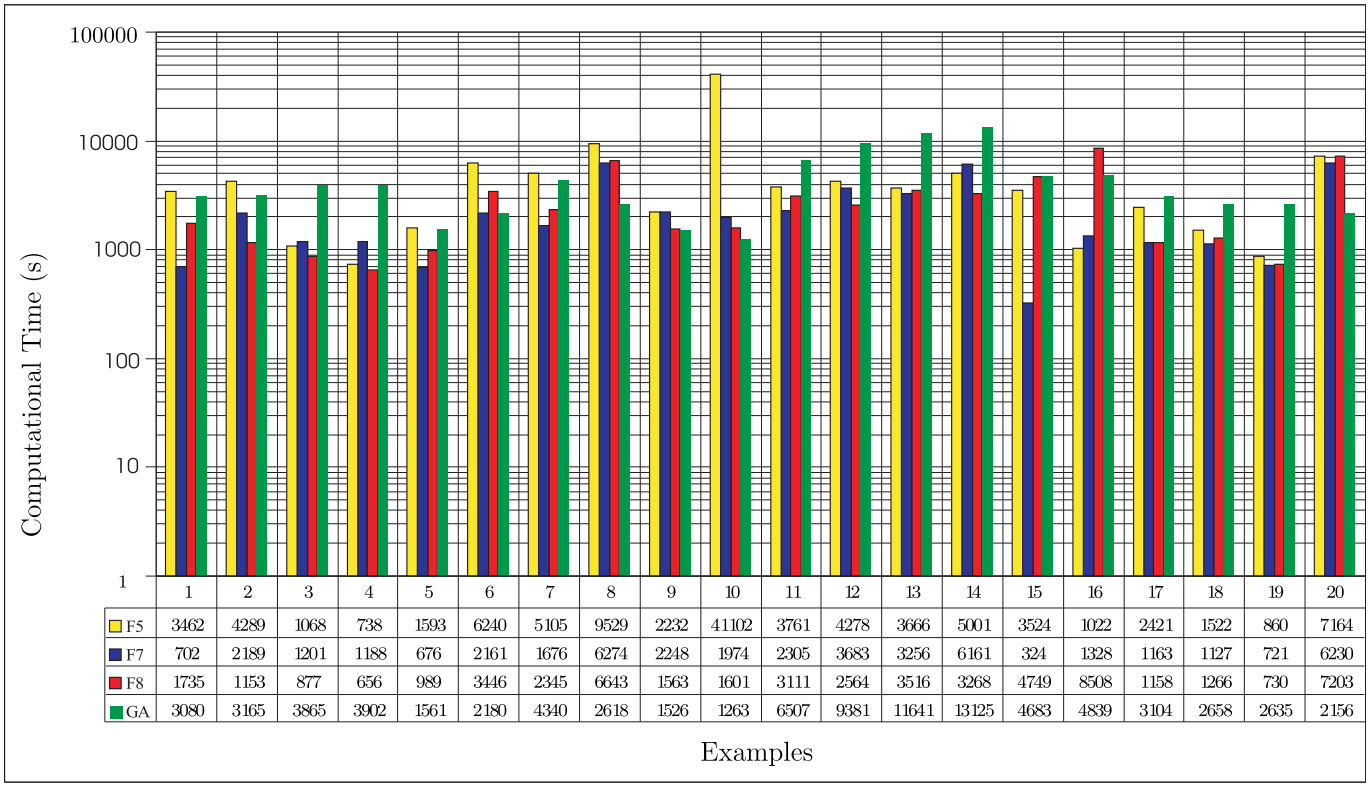
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F5	79	98	90	138	63	69	74	82	83	86	134	146	157	152	132	128	54	74	56	68
F7	64	91	73	87	65	65	85	79	60	67	105	98	135	141	94	73	63	71	61	56
F8	90	70	74	48	54	51	57	62	71	50	87	88	110	117	83	107	59	57	37	59
GA	0,9	1,1	6,3	6,2	1,2	1,3	1,3	0,7	0,9	0,8	1,1	1,3	4,4	3,6	1,7	1,3	1,3	1,9	0,9	1,1

Fig. 8. Second Group, Execution Time (s) Comparison with Rubio et al. (2010).



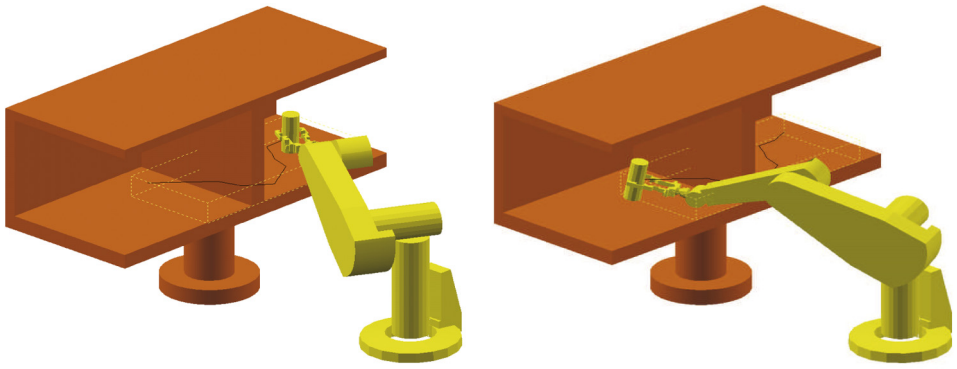
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F5	2,02	2,20	1,33	1,93	2,83	2,80	1,40	1,38	0,99	0,95	1,98	1,91	3,91	4,08	2,95	3,35	2,72	2,48	2,59	2,31
F7	2,31	2,23	1,46	1,62	2,55	2,41	1,75	1,45	1,01	1,04	2,08	1,98	3,66	4,29	3,36	2,70	2,73	2,24	2,76	2,11
F8	2,53	1,80	1,58	1,17	2,50	2,35	1,36	1,39	0,92	0,88	2,23	1,94	3,87	4,16	2,78	3,36	2,09	2,15	2,55	2,49
GA	1,63	1,64	1,60	1,75	2,28	2,40	1,33	1,21	0,88	0,85	1,66	1,56	3,55	3,57	2,20	2,09	1,40	1,90	1,86	1,91

Fig. 9. Second Group, Traveled Distance (m) Comparison with Rubio et al. (2010).



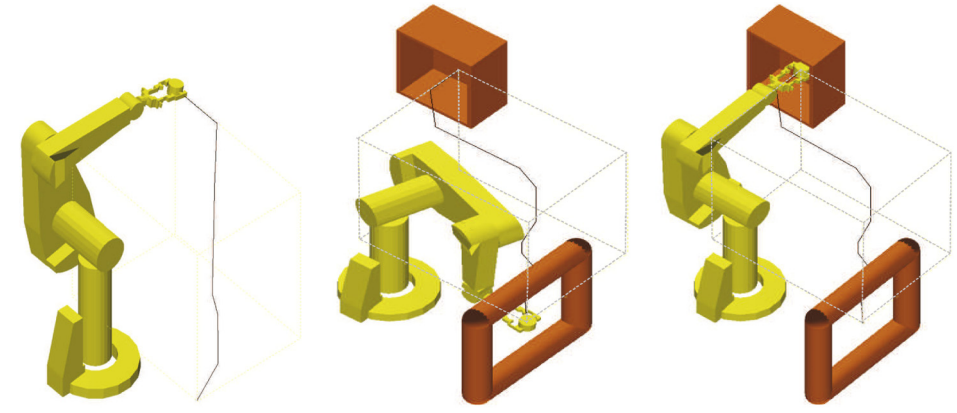
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F5	3462	4289	1068	738	1593	6240	5105	9529	2232	41102	3761	4278	3666	5001	3524	1022	2421	1522	860	7164
F7	702	2189	1201	1188	676	2161	1676	6274	2248	1974	2305	3683	3256	6161	324	1328	1163	1127	721	6230
F8	1735	1153	877	656	989	3446	2345	6643	1563	1601	3111	2564	3516	3268	4749	8508	1158	1266	730	7203
GA	3080	3165	3865	3902	1561	2180	4340	2618	1526	1263	6507	9381	11641	13125	4683	4839	3104	2658	2635	2156

Fig. 10. Second Group, Computational Time (s) Comparison with Rubio et al. (2010).



(a) Initial Configuration. (b) Final Configuration.

Fig. 11. Third Group, Industrial Application # 1.



(a) Case without Obstacles. (b) Initial Configuration. (c) Final Configuration

Fig. 12. Third Group, Industrial Application # 2.

7. Conclusion

In this paper, a new three-staged evolutionary algorithm for trajectory planning problem for industrial robots in connection with obstacle avoidance has been presented, in this case exemplarily shown for different workspace scenarios using a PUMA 560 six-axis manipulator. This method combines three formerly known evolutionary algorithms to solve the three stages, which are (1) acquisition of possible and collision-free configurations using SSGA, (2) path planning with minimum path length under kinematic constraint and (3) minimum time trajectory planning conditioned by the robot's dynamics. The main benefit of the presented method is its performance increase compared to common methods concerning execution time and traveling distance.

Working in a discrete configuration space implies generation of configurations that affecting the results. The offline trajectory planning of the robot was formulated using cubic spline functions.

A comparison of results has been established between the proposed algorithm and the algorithms introduced by Rubio et al. (2009, 2010). Two complete groups of numerical examples were presented and compared. In the first group, the execution time resulting from the GA procedure presented an average 6.21% of the values presented by Rubio et al. (2009, 2010), while the traveling distance presented 78.95% of the Rubio et al. (2009, 2010)'s values. In the second group, the execution time average is 2.35%, while the traveling distance average is 82.27%. Moreover, it can be seen that the Rubio et al. (2009)'s algorithm sometimes did not succeed in finding a solution, while the presented procedure always succeeded.

References

Abdel-Malek, K., Mi, Z., Yang, J., Nebel, K., 2006. Optimization-based trajectory planning of the human upper body. Robotica 24 (6), 683–696.

Abu-Dakka, F., March 2011. Trajectory Planning for Industrial Robot Using Genetic Algorithms (Ph.D. thesis). Departamento Ingeniera Mecnica y de Materiales, Universidad Politcnica de Valencia, Valencia, Spain.

Abu-Dakka, F., Valero, F., Mata, V., Assad, I., 2007a. Proc. for the 16th International Workshop on Robotics in Alpe-adria-danube Region. In: Path Planning Optimization of Industrial Robots Using Genetic Algorithm. Ljubljana, pp. 424–429.

Abu-Dakka, F., Valero, F., Tubaileh, A., Rubio, F., 2007b. Proc. of Twelfth World Congress in Mechanism and Machine Science IFToMM2007. In: Obtaining Adjacent Configurations with Minimum Time Considering Robot Dynamics. Besançon, France.

Abu-Dakka, F., Valero, F., Mata, V., 2008. Proc. for the 17th International Workshop on Robotics in Alpe-adria-danube Region. In: Obtaining Adjacent Configurations with Minimum Time Considering Robot Dynamics Using Genetic Algorithms. Ancona.

Aurelio, P., Visioli, A., 2000. Global minimum-jerk trajectory planning of robot manipulators. IEEE Transactions on Industrial Electronics 47 (1), 140–149.

Babu, B., Anbarasu, B., 2005. Proc. of Third International Conference on Computational Intelligence, Robotics, and Autonomous Systems (Ciras-2005). In: Multi-objective Differential Evolution (MODE): an Evolutionary Algorithm for Multi-objective Optimization Problems (MOOPs). Singapore.

Behzadipour, S., Khajepour, A., 2006. Time-optimal trajectory planning in cable-based manipulators. International Journal for Numerical Methods in Engineering 22 (3), 559–563.

Bertolazzi, E., Biral, F., Lio, M., 2007. Real-time motion planning for multibody systems: real life application examples. Multibody System Dynamics 17 (2–3), 119–139.

Chettibi, T., Lehtihet, H., Haddad, M., Hanchi, S., 2004. Minimum cost trajectory planning for industrial robots. European Journal of Mechanics – A/Solids 23 (4), 703–715.

Craig, J., 2005. Introduction to Robotics: Mechanics and Control, third ed. Prentice Hall.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.

Elnagar, A., Hussein, A., 2000. On optimal constrained trajectory planning in 3d environments. Robotics and Autonomous Systems 33 (4), 195–206.

Fonseca, C., Fleming, P., 1995. An overview of evolutionary algorithms in multi-objective optimization. Evolutionary Computations Journal 3 (1), 1–16.

Gasparetto, A., Zanotto, V., 2007. A new method for smooth trajectory planning of robot manipulators. Mechanism and Machine Theory 42 (4), 455–471.

Horn, J., Nafploitis, N., Goldberg, D., 1994. Proc. of the First IEEE Conference on Evolutionary Computation. In: A Niched Pareto Genetic Algorithm for Multi-objective Optimization. Orlando, FL, pp. 82–87.

Lin, C., Chang, P., Luh, J., 1983. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. IEEE Transactions on Automatic Control 28 (12), 1066–1073.

Piazzi, A., Visioli, A., 1997. Proc. of IEEE-RSJ International Conference on Intelligent Robots and Systems. Grenoble, pp. 1553–1559.

Piazzi, A., Visioli, A., 2000. Global minimum-jerk trajectory planning of robot manipulators. IEEE Transactions on Industrial Electronics 47 (1), 140–149.

Plessis, L.D., Snyman, J., 2003. Trajectory-planning through interpolation by overlapping cubic arcs and cubic splines. International Journal for Numerical Methods in Engineering 57 (11), 1615–1641.

Rubio, F., Valero, F., ner, J.S., Mata, V., 2009. Simultaneous algorithm to solve the trajectory planning problem. Mechanism and Machine Theory 44 (10), 1910–1922.

Rubio, F., Valero, F., ner, J.S., 2010. The simultaneous algorithm and the best interpolation function for trajectory planning. Industrial Robot: An International Journal 37 (5), 441–451.

Saramago, S., Ceccareli, M., 2002. Trajectory modelling of robot manipulators in the presence of obstacles. Robotica 20 (4), 395–404.

Saramago, S., Steffen, V., 1998. Optimization of the trajectory planning of robot manipulators taking into-account the dynamics of the system. Mechanism and Machine Theory 33 (7), 883–894.

Saramago, S., Steffen, V., 1999. Dynamic optimization for the trajectory planning of robot manipulators in the presence of obstacles. Journal of the Brazilian Society of Mechanical Sciences 21 (3), 1–17.

Saramago, S., Steffen, V., 2000. Optimal trajectory planning of robot manipulators in the presence of moving obstacles. Mechanism and Machine Theory 35 (8), 1079–1094.

Saramago, S., Steffen, V., 2001. Trajectory modelling of robot manipulators in the presence of obstacles. Journal of Optimization Theory and Applications 110 (1), 17–34.

Saravanan, R., Ramabalan, S., Balamurugan, C., 2009. Evolutionary multi-criteria trajectory modeling of industrial robots in the presence of obstacles. Engineering Applications of Artificial Intelligence 22 (2), 329–342.

Valero, F., Mata, V., Cuadrado, J., Ceccarelli, M., 1996. A formulation for path planning of manipulators in complex environments by using adjacent configurations. Advanced Robotics 11 (1), 33–56.

Valero, F., Mata, V., Besa, A., 2006. Trajectory planning in workspaces with obstacles taking into account the dynamic robot behaviour. Mechanism and Machine Theory 41 (5), 525–536.

Wall, M., 1996. Galib: a C++ Library of Genetic Algorithm Components. URL. http://lancet.mit.edu/ga.

Wang, C., Horng, J., 1990. Constrained minimum-time path planning for robot manipulators via virtual knots of the cubic b-spline functions. IEEE Transactions on Automatic Control 35 (35), 573–577.