Comparing the efficiency of five algorithms applied to path planning for industrial robots

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

CITV, Universidad Politécnica de Valencia, España.
e-mail: frubio@mcm.upv.es

Abstract

Purpose- To compare the quality and efficiency of five methods for solving the path planning problem of industrial robots in complex environments

Design/methodology/approach- Five methods are presented for solving the path planning problem and certain working parameters have been monitored using each method. These working parameters are the distance travelled by the robot and the computational time needed to find a solution. A comparison of results has been analyzed.

Findings-After this study, it could be easy to know which of the proposed methods is most suitable for application in each case, depending on the parameter the user wants to optimize. The findings have been summarized in the conclusion section.

Research limitations/implications- Five techniques have been developed which yield good results in general.

Practical implications- The algorithms introduced are able to solve the path planning problem for any industrial robot working with obstacles.

Social implications- The path planning algorithms help robots perform their tasks in a more efficient way due to the fact that the path followed has been optimized and therefore they help humans work together with the robots in order to obtain the best results from them.

Originality/value- The paper shows which algorithm offers the best results depending on the example the user has to solve and the parameter to be optimized.

Keywords*:* Path Planning; Collision avoidance; Off-line Programming; Kinematics; Multi-arms robot;

1. Brief introduction

It is a fact that the number of robots in industry has increased year after year worldwide and it is estimated that the percentage will increase in the coming years. This fact makes us reflect on the importance of robotics, not only in its application to production processes but also in the field of research and innovation.

Although the most important sector is the automotive one, the robot has spread to others such as services, construction, medicine, food, home automation, defense, aerospace, etc.

The ultimate goal is to increase productivity, reduce accidents and enhance quality and flexibility. In this context the importance of industrial robot programming should be emphasized. Path planning is a tool that helps the robot to perform the required tasks efficiently.

In recent years there have been many researchers that in one way or another have contributed to the development of techniques and methods to solve the motion problem and in particular the path planning problem.

It´s not an easy task to make a list of the most important events that have marked the evolution of path planning over the years since there is the risk of overlooking researchers who have made a contribution. In any case, in the following paragraphs some of those studies that are widely recognized by the scientific community will be mentioned.

Nilsson addressed the problem of finding the shortest path in large workspaces by introducing heuristic search techniques (Nilsson *et al*., 1968). Lozano-Perez introduced the concept of configuration space (Lozano-Perez, 1983). That same year he introduced the method of approximate cell decomposition (together with Brooks, Brooks *et al.*, 1983). O'Dúnlaing and Yap introduced the method of shrinkage. Their approach is based on the use of the Generalized Voronoi diagram (Ó'Dúnlaing *et al.*, 1985).

Moreover, the work done by Brooks, 1983, Schwartz *et al.*, 1983, Dubowsky *et al.*, 1984 and Khatib, 1985 in the eighties should not be forgotten which leads to those methods based on probabilistic planners (PRMs) (Kavraki, 1994 and Hsu, 2002), where a graph called the probabilistic roadmap is obtained. Afterwards, the roadmap is analyzed (Kavraki, 1996 and 1998) in search of the path that connects the initial and final configuration. Other probabilistic techniques for solving the path planning problem have also appeared (Lavalle, 2000), for example the algorithm based on Rapidly-Exploring Random Trees (RRTs). The planner proposed by Valero *et al.*, 1997, in which all the workspace is analysed, is also worth mentioning. Besides the methods already mentioned, genetic algorithms have also been implemented successfully for the path planning problem (see Monteiro *et al.*, 1999, Davidor, 1991 and Yang *et al.*, 2008.

There are many authors who have made a historical review of these algorithms over the years. Some examples are Kavraki and Latombe, 1998, Isto, 1996, LaValle, 1996 and 2000, Gupta, 1998, etc. a historical approach to genetic algorithms is presented by Fares, 2007.

Rubio *et al.*, 2009 presented an approach in which the search of the path is made in the state space of the robotic system (direct approach), and it makes use of the information generated about the characteristics of the process and the techniques for introducing a branching graph.

In previous studies by this author (Rubio, 2006), it can also be found the most recent records used to develop the path planning algorithms that we present in this paper. This paper will analyze the results obtained using the five methods. These are:

A. Indirect algorithm, also called "sequential" because it calculates the path sequentially: first it generates the subspace of the robot´s configuration space by searching the discretized workspace, and then it eventually finds an optimal path in this space.

B. Direct algorithm, also called "simultaneous". This algorithm analyzes the configuration space of the robotic system. It is characterized by finding a solution at the same time as the robot reaches the goal, that is, when it reaches the final configuration. While the robot is moving to the final configuration it will simultaneously generate a path. Three variants have been introduced, depending on the characteristics of the cost function that is used to guide the search of the path:

Option 1: Cost function A*

Option 2: Uniform Cost Function

Option 3: Greedy Cost function

C. Genetic algorithm, based on the characteristics of genetic algorithms applied to robotics.

The five algorithms have been applied to the same PUMA 560 robot, which has been modelled in wired format, see Rubio *et al.*, 2009. To obtain collision-free configurations, obstacles have been modelled using patterned obstacles: spheres, cylinders and planes (see also Rubio *et al.*, 2009).

In the following sections the most important points related to the mentioned algorithms will be discussed. Special attention should be paid to the application examples section.

2. Robot modelling and collision avoidance

The robotic system is common to the different algorithms. It is represented using its wired model. Any robot configuration $C^j = C^j(\alpha_i^j, p_k^j)$ can be expressed unequivocally by means of the Cartesian coordinates of significant points of the robot $\alpha_i^j = (\alpha_{xi}^j, \alpha_{yi}^j, \alpha_{zi}^j)$. Any other point of interest in the robot, p_k^j, will also be expressed in Cartesian coordinates, i.e, $p_k^j = (p_{xk}^j, p_{yk}^j, p_{zk}^j)$, calculated from the significant points, namely, $p_k^j = p_k^j(\alpha_i^j)$.

For the PUMA 560 robot, seven significant points are used (α_1, α_2, α_3 ,α_4, α_5, α_6 and α_7) along with five additional points of interest ($p_1 \ldots p_5$) called auxiliary points which are calculated from the significant points (see Fig. 1). Cartesian coordinates are justified to calculate distances and avoid collisions.

The genetic algorithm uses both joint coordinates and Cartesian coordinates and the rest of the algorithms only use Cartesian coordinates.

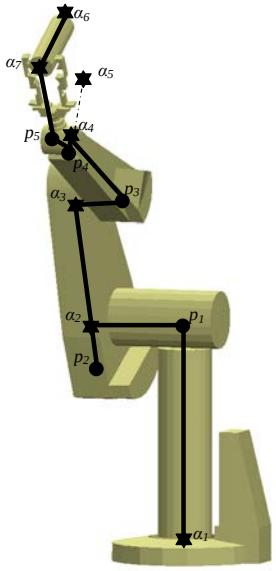


Figure 1 Robot and its wired model
Note: Significant points and points of interest

To prevent collisions, three different and basic patterned shapes of obstacles have been used: spheres, cylinders and prisms. Any real obstacle can be modelled by adding patterned obstacles. The robot links are also modelled as cylinders so as to avoid collisions the algorithm control the distances between the robot´s links and the obstacles. Distances are constraints in the optimization problem so the algorithms calculate collision-free adjacent configurations (see Section 7). Further details on robot modelling and collision avoidance can be found in Valero *et al.*, 1997 and Rubio *et al.*, 2009.

3. Generation of Adjacent Configurations.

Configuration space is obtained by generating adjacent configurations (see Rubio *et al.*, 2009). Given a feasible configuration C^k, it is said that a new configuration C^p is adjacent to it if it is also feasible (i.e. it meets the characteristics of the robot while avoiding collisions), and in addition the following three properties are fulfilled:

1. The position of the end-effector of configuration C^p is at a distance of one unit with respect to the position of the end-effector of configuration C^k.
2. There are no obstacles between adjacent configurations C^k and C^p.
3. Configuration C^p should minimize this objective function:

$$\left\| C^p - C^f \right\| = \sum_{i=1}^{n} \left(\left(\alpha_{xi}^p - \alpha_{xi}^f \right)^2 + \left(\alpha_{yi}^p - \alpha_{yi}^f \right)^2 + \left(\alpha_{zi}^p - \alpha_{zi}^f \right)^2 \right) \quad (1)$$

where C^f is the final configuration

4. An improved method to calculate distances

By controlling the distance from the different patterned obstacles to the cylinders that cover the robot links, collision avoidance between the robot and the obstacles is possible. Distances are constraints in the optimization problem. They serve to calculate collision-free adjacent configurations.

The application of the procedure between link *i* of the robot, modelled as a cylinder $RC_i = \left(c_{1i}^{RC}, c_{2i}^{RC}, r_i^{RC} \right)$ and the patterned obstacle *j* (which may be a cylinder, sphere or planar surface), can give rise to three different cases to prevent collisions:

A) Cylinder-Sphere:

Here we compute the distance between a line segment (cylinder) to a point (centre of the sphere). Let *AB* be a line segment specified by the endpoints *A* and *B*. Given an arbitrary point *C*, the problem is to determine the point *P* on *AB* closest to *C*. Then we calculate the distance between these two points.

Projecting *C* onto the extended line through *AB* provides the solution. If the projection point *P* lies within the segment, then *P* itself is the correct answer.

If *P* lies outside the segment, then the segment endpoint closest to *C* is instead the closest point (A or B). See Fig. 2.

B) Cylinder-Cylinder

Here we compute the distance between two line segments.

The problem of determining the closest points between two line segments S_1 (P_1Q_1) and S_2 (P_2Q_2) (and therefore the distance) is more complicated than computing the closest points of the lines L_1 and L_2 of which the segments are a part. Only when the closest points of L_1 and L_2 happen to lie on the segments does the method for closest points between lines apply. For the case in which the closest points between L_1 and L_2 lie outside one or both segments, a common misconception is that it is sufficient to clamp the outside points to the nearest segment endpoint. It can be shown that if just one of the closest points between the lines is outside its corresponding segment, that point can be clamped to the appropriate endpoint of the segment and the point on the other segment closest to the endpoint is computed.

If both points are outside their respective segments, the same clamping procedure must be repeated twice. See Fig. 2

C) Cylinder-Planar Surface.

The surface is divided into triangles. In this case we compute the distance between a line segment and a triangle. See Fig. 2.

The closest pair of points between a line segment *PQ* and a triangle is not necessarily unique. When the line segment is parallel to the plane of the triangle, there may be an infinite number of equally close pairs. However, regardless of whether the segment is parallel to the plane or not, it is always possible to locate a point such that the minimum distance falls either between the end point of the segment and the interior of the triangle or between the segment and an edge of the triangle. Thus, the closest pair of points (and therefore the distance) can be found by computing the closest pairs of points between the following entities:

-Segment *PQ* and triangle edge *AB*.

-Segment *PQ* and triangle edge *BC*.

-Segment *PQ* and triangle edge *CA*.

-Segment endpoint *P* and plane of triangle (when *P* projects inside *ABC*)

-Segment endpoint *Q* and plane of triangle (when *Q* projects inside *ABC*).

The number of tests required to calculate the distance can be reduced in some cases.

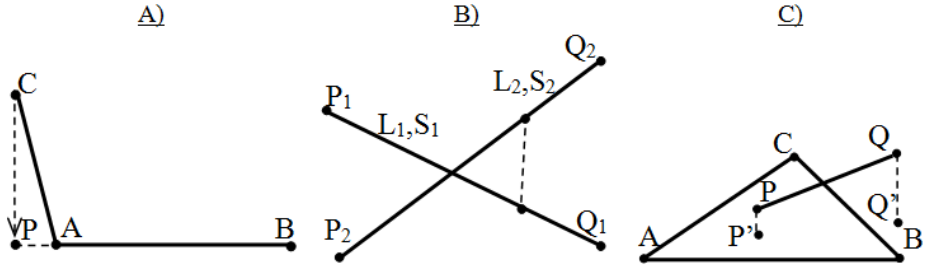


Figure 2 Three different cases to prevent collisions

5. Characteristics of the five algorithms

5.1 Sequential algorithm

The path consists of a sequence of configurations, which are between the initial and final ones. The set of intermediate configurations that are part of the path, so that the robot goes from one place to another, is calculated sequentially. In the first part, the configuration space is generated from the initial configuration by calculating adjacent configurations. The robot end-effector must visit all the nodes that are part of the discretized workspace (see Rubio *et al.*, 2009) until the final configuration is reached. Having calculated the configuration space, a weighted graph is created. Each node in the graph stores a configuration of the robot, and the arcs between nodes represent the distance between adjacent configurations.

The weight corresponding to the arc that goes from node k (C^k robot configuration) to node p (C^p robot configuration), can be given as:

$$a(k, p) = \sum_{i=1}^{n} \left(x_i^p - x_i^k\right)^2 \tag{2}$$

where that C^k and C^p are adjacent configurations, n is the number of Cartesian coordinates used in the wired model and x_i^p and x_i^k are the Cartesian coordinates of the significant points corresponding to configurations C^k and C^p respectively. In addition C^k and C^p must satisfy:

1. Collision avoidance within the robot workspace therefore Constraints type (d) avoiding the obstacles between configurations;
2. Constraints on the mobility of robot joints, takink into account the limits to each actuator´s motion.

If the configurations are not adjacent (one of the properties defined in the previous section is not met) the weight is considered as

$$a(k, p) = \infty \tag{3}$$

In the second part, the searching is started in the weighed graph with the path that joins the node corresponding to the initial configuration to the node corresponding to the goal configuration. The aim is to join the initial and final configuration through a path (sequence of configurations) of minimum distance. Since the arcs meet that $a(k,p) \geq 0$, the Dijkstra's algorithm is used in order to obtain the path that minimizes the distance between the initial and goal configurations. If this path exists, that means there is a solution, the algorithm can find the sequence of configurations that must follow the robot.

5.2 Simultaneous algorithm and its variants

This algorithm has three variants depending on the cost function used. This algorithm simultaneously calculates the configuration space and the shortest path as the algorithm evolves looking for the solution. It is necessary to define the objective function to guide the search for new adjacent configurations. The objective function is defined in terms of the distances that the robot must travel between configurations. To calculate the distances it is necessary to attach a node to each configuration of the robot. Initially there are only nodes corresponding to the initial and final configurations. Then a graph is generated that will be completed as new configurations are obtained. Each node in the graph represents a possible configuration of the robot, and the arcs between nodes represent the distance traveled by the significant points of the robot between these configurations.

The branching process that creates new adjacent configurations from a given one is as follows: first adjacent configurations are obtained from the current node; secondly, the adjacent configurations generated are evaluated using a cost function associated with the distances travelled by the significant points of the robot. This cost function is different from the objective function used in the generation of adjacent configurations. Finally, we evaluate the new nodes obtained and the branching process will continue from that which is most promising, i.e. that which presents a lower cost function.

We present three different cost functions to evaluate the branch nodes, therefore it gives three different algorithms:

2.1. Option 1: The cost function associated with the algorithm A *. It corresponds to the uniform cost function augmented by the estimated cost of moving from the current configuration to the final. If the node *p* (corresponding to the configuration adjacent to C^p) has been generated from the node *k* (configuration C^k), the cost function *f (p)* associated with the algorithm A * can be broken down as follows:

$$f(p)= f(k)+ c(k,p)+ h(p,f) \tag{6}$$

The term *f (k)* is indicative of the cost of reaching the node k, and it is a known value (since configuration C^k has already been reached).

The term *c (k, p)* is indicative of the cost of moving from an adjacent configuration C^k to another C^p . It is obtained:

$$c(k,p)=\sqrt{\left(\sum_{i=1}^{n}\left|\alpha_i^p-\alpha_i^k\right|^2\right)} \tag{7}$$

where *n* is the number of significant points of the robot.

The term *h (p, f)* corresponds to an estimate of the cost of moving from the configuration adjacent C^p to the final configuration C^f. It is calculated:

$$h(p,f)=\sqrt{\left(\sum_{i=1}^{n}\left|\alpha_i^p-\alpha_i^f\right|^2\right)} \tag{8}$$

where $n = 7$, C^p is the adjacent configuration generated and C^f is the final configuration.

2.2. Option 2: Uniform cost function. If the node *p* (corresponding to the configuration adjacent C^p) has been generated from the node *k* (C^k), the cost function *f (p)* associated with node *p*, can be broken down as follows:

$$f(p)= f(k)+ c(k,p) \tag{4}$$

The term *f (k)* is indicative of the cost of reaching node *k* from the initial configuration. Its value is known as configuration C^k has been reached. It is calculated recursively.

The term *c (k, p)* is indicative of the cost of moving from an adjacent configuration C^k and C^p. It is obtained:

$$c(k,p)=\sqrt{\left(\sum_{i=1}^{n}\left|\alpha_i^p-\alpha_i^k\right|^2\right)} \tag{5}$$

where α_i^p and α_i^k are the significant points of configurations C^p and C^k respectively, and $n = 7$ is the number of significant points of the robot.

2.3. Option 3: The cost function associated with a greedy search: it only takes into account the estimated cost of moving from the actual configuration to the final configuration or objective. The total cost function associated with the greedy algorithm can be highlighted as follows:

$$f(p)= h(p,f) \tag{9}$$

h (p, f) representing the estimated cost of going from configuration C^p to the final configuration C^f .

Knowing the cost function implemented in the algorithm, it is possible to branch out. First, a node has to be selected from all the nodes generated (corresponding to new configurations). The one that raises expectations of producing the optimal solution (shortest path) is selected. Thus, among the nodes accumulated so far which have not yet been branched out, the node whose cost function is the smallest for further branching will be chosen.

This process is repeated indefinitely, branching out the nodes obtained until reaching the final configuration. Simultaneously with this branching process, the path followed is stored, until reaching the goal node (or final configuration). When the above process has come to an end, we will also have the sequence of valid configurations between the initial and final configuration and therefore a solution to the path planning problem. Most of the time, it is not necessary to evaluate the entire workspace to obtain a solution.

5.3 Genetic algorithm

GAs are search algorithms based on mechanics of natural selection and natural genetics. They combine survival of the most fitting among the string structures with randomized yet structured information exchange to form a search algorithm with innovative flair of natural evolution.

They have recently found an increasing use in machine learning, robot motion planning, scheduling, pattern recognition, image sensing and many other engineering applications. GAs are search algorithms based on mechanics of natural selection and natural genetics. They combine survival of the most fitting among the string structures with randomized yet structured information exchange to form a search algorithm with innovative flair of natural evolution. In this paper a genetic algorithm is applied to path planning to obtain a path of minimum distance from two configurations (initial and final) given by the user.

The genetic algorithm (GA) solves such problem by minimizing the traveling distance of the end-effector and the significant points between the initial and final configurations avoiding obstacles. The workspace will be modeled in such way to provide a discrete configuration space based on the positions of the end-effector between the initial and final configurations of the robot.

In this procedure, two optimization processes using genetic algorithms are involved. The first one is an optimization process for obtaining the adjacent configurations. The order in which the adjacent configurations are generated will condition the Space of Configurations generated and, therefore, the path to be obtained. The second optimization process is used for the obtaining of the path, which consist of a set of adjacent configurations. This algorithm is applied on an industrial robot Puma 560 modeled with six degree of freedom.

5.3.1 Obtaining the Configuration C^k

In the building process of the path, a random search procedure will be applied to search from the C^i for the next adjacent configuration and so on until it reaches the C^f. The main concern in this part is finding a sequence of robot configurations between the initial and final configurations that fulfils the early listed three conditions.

A Steady State Genetic Algorithm (SSGA) procedure is used to obtain a robot configuration C^k adjacent to a given one C^P.

The individual or the chromosome represents the robot configuration. Each chromosome consists of six genes; the robot generalized coordinates (q_i; i = 1, 2, …, 6).

$$Individual = \{q_1, q_2, q_3, q_4, q_5, q_6\} \tag{10}$$

The initial population consists of a defined number of individuals. The initial values of each gene in the individual are selected randomly between the two limits of the generalized coordinates for that gene.

A roulette-wheel selection method is applied to select individuals for crossover and mutation. This method is 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.

The crossover operator defines the procedure for generating a child from two selected parents. A single point crossover is used in this procedure. See Fig. (3).

The mutation operator defines the procedure for mutating each genome. In this procedure, one gene mutation is used. An offspring will be selected randomly and then a gene will be selected randomly from that offspring. Afterwards, a gene will be mutated

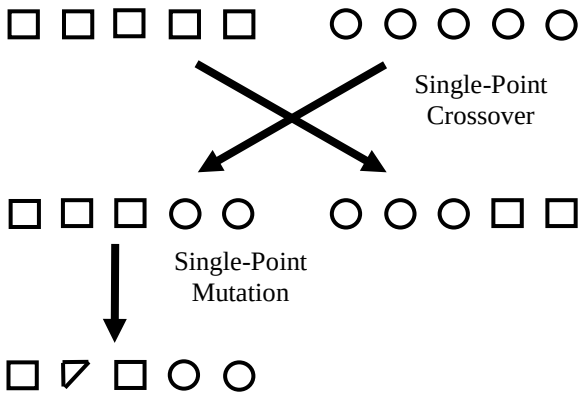


Figure 3 Adjacent configuration crossover and mutation

Using mechanisms of crossover and mutation, a new generation of solutions is obtained from which a subset is selected to form the basis for creating a new generation that is characterized by the fact that its properties (distance travelled) have been improved. This is an iterative process that continues until the optimal solution has been found acceptable when the number of generations has reached a predetermined level.

The objective of this GA procedure is to minimize the distance between significant points of the two adjacent configurations in order to find the optimal path between the initial and final configurations of the robot. The shortest path will be calculated by minimizing the sum of the straight line segments of the corresponding significant points of the robot from the initial to the final configuration.

$$Minimize\left\{\sum_{i=1}^{n-1}\sum_{j=1}^{m}\sqrt{\left(\alpha_j^{i+1}-\alpha_j^{i}\right)_x^2+\left(\alpha_j^{i+1}-\alpha_j^{i}\right)_y^2+\left(\alpha_j^{i+1}-\alpha_j^{i}\right)_z^2}\right\} \tag{11}$$

where: j is the number of significant points of the robot, and $m = 4$ for the robot Puma 560 (when using this algorithm) and $i = 1, 2, \ldots, n$ is the number of robot configurations included in the path.

5.3.2 Obtaining the path

The genetic algorithm for path planning uses parallel populations with migration technique. The genetic algorithm has multiple, independent populations. The master population is updated each generation with best individual from each population.

In the GA based solution procedure, a number of new individuals are created at each iteration. The remaining individuals are obtained by deterministically copying the individuals with the top fitness from the previous generation.

The individual or the chromosome is composed of set of intermediate points (end-effector positions) including end points (initial and final position of the end effector). This means that each chromosome represents a complete path between initial and final configurations.

The first point of each individual is the initial position of the end-effector of C i. The second point will be selected randomly from the discretized workspace without repetition in one of seven directions: X-, Y-, Z-, XY-, XZ-, YZ-, and XYZ-direction. This strategy will be repeated for the next point and so on until the goal position is achieved. Note that this definition is based on the number of intermediate points that constitute the path, which means that paths do not have equal lengths, which leads to more complexity in crossover and mutation.

The objective of this optimization problem is to find the optimal path between initial and final positions of a robot end-effector by minimizing the sum of the straight-line segments of the corresponding significant points of the robot, from the initial to the final point.

The selection operation is made using the roulette-wheel method.

The crossover is made through the exchange of a part of the path (chromosome) between two selected paths through the selection operation mentioned earlier; being that, it is executed only if the probability of the crossover is satisfied. For more details please refer to Fares, A.D. et al. (2007).

The mutation is done by selecting a configuration (gene) randomly from a selected path (chromosome). The first and the final configurations are not considered for mutation. For more details please refer to Fares, A.D. et al. (2007).

6. Comparing the efficiency of the algorithms

6.1 Introduction

The examples that appear in this section have been solved using an Intel (R) Core (TM) 2 Quad computer with 2.83 GHz and 4 GB of RAM.

We are going to analyze the quality and efficiency of the five proposed algorithms: the algorithm called sequential (referred to as SEQ), the simultaneous algorithm with cost function A* (referred to as A*), the simultaneous algorithm with uniform cost function (referred to as UC), the simultaneous algorithm with greedy cost function (referred to as G) and the genetic algorithm (referred to as GA). We will also compare the results between them.

To do this study we apply the algorithms on a set of eleven examples and we will monitor two important operating parameters: the computational time required to generate a solution and the distance travelled by the robot.

6.2 Examples

A set of eleven examples has been solved using the PUMA 560 robot (with and without obstacles), and up to three sets of differently patterned obstacles in each example have been introduced sequentially: the first one a spherical obstacle, then a second cylindrical obstacle interfering with the previous path and requiring the calculation of a new path for the algorithm and a third prismatic obstacle trying to obstruct as much as possible the new previous path obtained. In each example the robot tends to go from the initial C^{i} to the final C^{f} configuration. In general, obstacles in the workspace are modeled using patterned obstacles: spheres, cylinders and prims.

Each of the eleven examples starts from different initial and final configurations and were solved initially by considering that there were no obstacles in the environment. All the algorithms developed generate a solution with their own properties (mainly distance travelled and computational time employed). Once we know the path to be followed by the robot´s end-effector, the first obstacle (sphere) was introduced, which hinders the realization of that path. Then the problem with this new obstacle is solved again, and new solutions are found that prevent the robot´s passage through the area occupied by the obstacle. The properties of new solutions obtained with different algorithms are analyzed again. With the new path calculated for the second time, a second obstacle (a cylinder) is introduced which hinders the realization of that path. With the new obstacle´s data, the path planning problem was solved again (now with two obstacles) and a new path was found that prevents collision with the obstacles. Finally, we introduce a third obstacle (a prism) which hinders the previous path and the algorithms look for a new path and they find it.

Having found the new solution, its properties are analyzed. The characteristics of the different paths have been recorded and displayed in the corresponding graphs.

6.3 Algorithms and obstacles used to solve the examples

Each example has been resolved following this system:

Example	Obstacles	Algorithm
nº *i* (*i=1.. 11*)	1 With 0 obstacles in the work environment	1.1. Sequential algorithm (SEQ). 1.2. Simultaneous algorithm A* (A*). 1.3. Simultaneous algorithm with Uniform cost (UC). 1.4. Simultaneous algorithm with Greedy cost (G). 1.5. Genetic algorithm (GA).
	2 With 1 obstacles in the work environment (spherical obs.)	2.1. Sequential algorithm (SEQ). 2.2. Simultaneous algorithm A* (A*). 2.3. Simultaneous algorithm with Uniform cost (UC). 2.4. Simultaneous algorithm with Greedy cost (G).

		2.5. Genetic algorithm (GA).
	3 With 2 obstacles in the work environment (spherical and cylindrical obs.)	3.1. Sequential algorithm (SEQ). 3.2. Simultaneous algorithm A* (A*). 3.3. Simultaneous algorithm with Uniform cost (UC). 3.4. Simultaneous algorithm with Greedy cost (G). 3.5. Genetic algorithm (GA).
	4 With 3 obstacles in the work environment (spherical, cylindrical and prismatic obs.)	4.1. Sequential algorithm (SEQ). 4.2. Simultaneous algorithm A* (A*). 4.3. Simultaneous algorithm with Uniform cost (UC). 4.4. Simultaneous algorithm with Greedy cost (G). 4.5. Genetic algorithm (GA).

The information generated during the resolution process is stored. What has mainly been analyzed is the computational time needed to obtain each solution and the distance travelled by the significant points of the robot. With this information, graphs 1 to 4 have been prepared.

Example nº 1

Next we describe in detail the process for the resolution of example n º 1 using two algorithms: the simultaneous one with greedy cost function (G) and the genetic algorithm (GA).
Fig. 4 shows the robot with the initial C^i and final C^f configurations.

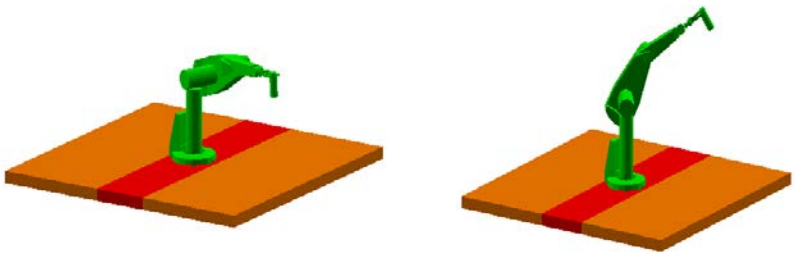


Figure 4 Initial and final configurations of the robot

Table 1 shows the joint coordinates for the PUMA 560 robot for initial C^i and final C^f configurations.
Table 2 shows the locations of the obstacles placed in the workspace for this example.

Table 1: Configurations to be joined (example nº 1)

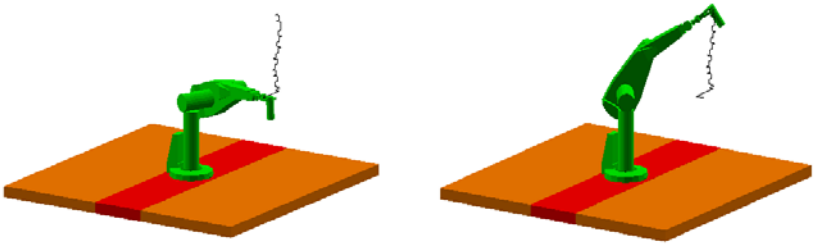
Joint Nº	Initial Configuration	Final Configuration
1	-23.79º	35.02º
2	-21.69º	-60.82º
3	106.18º	122.27º
4	-0.11º	162.92º
	0.0º	4.51º
6	0.0º	-159.88º
7	0.0 mm	0.0 mm

Table 2: Obstacle Location (in m.) (example nº 1)

1 obst.	Sphere	$c_1^{SO} = (-0.8, -0.3, 1.0)$		$r_1^{CO} = 0.15$	
2 obst.	Cylinder	$c_{11}^{CO} = (0.7, 0.3, 0)$	$c_{21}^{CO} = (0.7, 0.3, 1.1)$	$r_1^{CO} = 0.15$	
3 obst.	Prismatic	$a_3^{PO} = (-0.1, 0.1, 1.5)$	$q_{13}^{PO} = (-0.1, 0.1, 1.7)$	$q_{23}^{PO} = (0.5, 0.1, 1.5)$	$q_{33}^{PO} = (-0.1, -0.2, 1.5)$

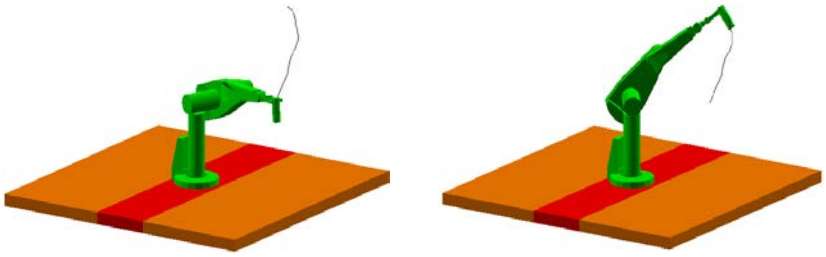
(The sphere is defined using the Cartesian coordinates of its centre c_1^{SO} and the radius r_1^{SO}. The cylinder is defined using the Cartesian coordinates of the centres of its bases c_{11}^{CO} and c_{21}^{CO} and the radius r_1^{CO}. The prismatic obstacle is defined using the Cartesian coordinates of four points corresponding to its vertexes a_3^{PO}, q_{13}^{PO}, q_{23}^{PO} and q_{33}^{PO}).

Fig 5 and Fig. 6 shows the solution obtained with algorithm G and GA respectively when no obstacles have been considered in the workspace.



Initial Configuration Final Configuration

Figure 5 Path obtained using algorithm G



Initial Configuration Final Configuration

Figure 6 Path obtained using algorithm GA

A spherical obstacle is then introduced into the workspace, trying to burden the path obtained previously. This is shown in Fig. 7 for algorithm G.

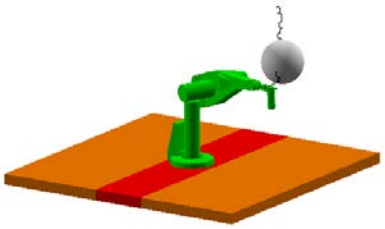
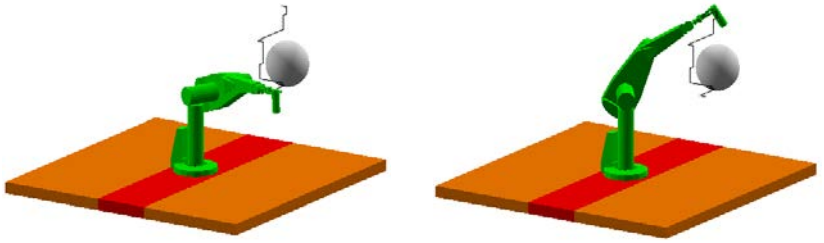


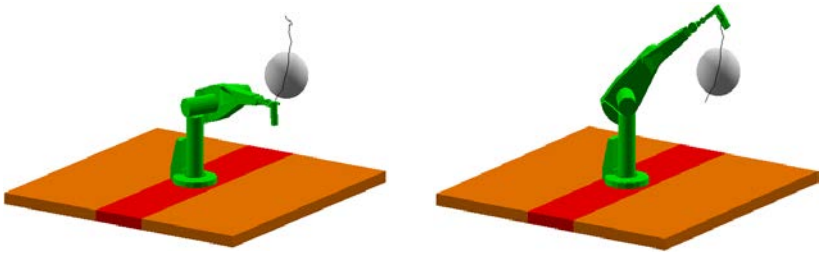
Figure 7 Obstacle introduced obstructing the path (algorithm G)

In Fig. 8 and Fig. 9 we can see the new solutions (new paths). We can observe how the new paths avoid the obstacle:



Initial Configuration Final Configuration

Figure 8 New path obtained using algorithm G



Initial Configuration Final Configuration

Figure 9 New path obtained using algorithm GA

Next, a second cylindrical obstacle is introduced, trying to obstruct the path obtained. This is shown in Fig. 10 for algorithm G.

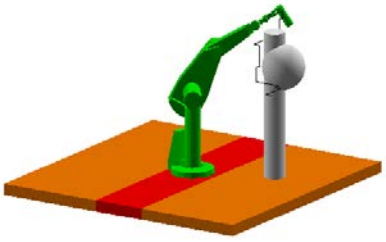
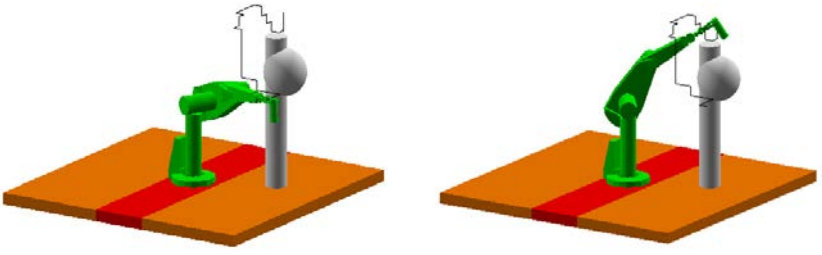


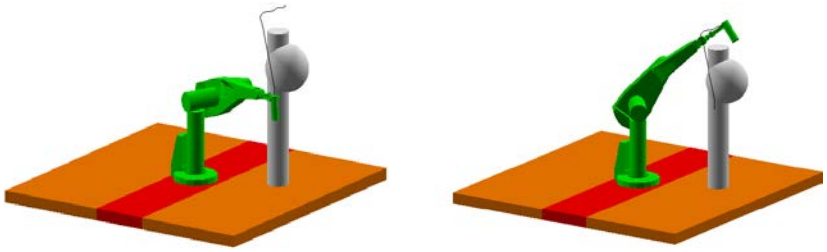
Figure 10 Second obstacle introduced, obstructing the previous path (algorithm G)

The problem is solved again. The new solutions obtained using algorithms G and GA are shown in Fig. 11 and 12.



Initial Configuration Final Configuration

Figure 11 New path obtained using the algorithm G



Initial Configuration Final Configuration

Figure 12 New path obtained using the algorithm GA

Finally, the third obstacle (a prism) is introduced. See Fig. 13.

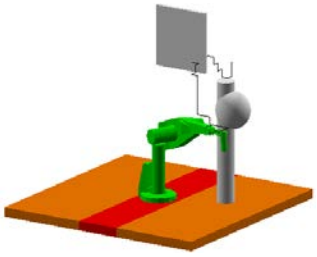
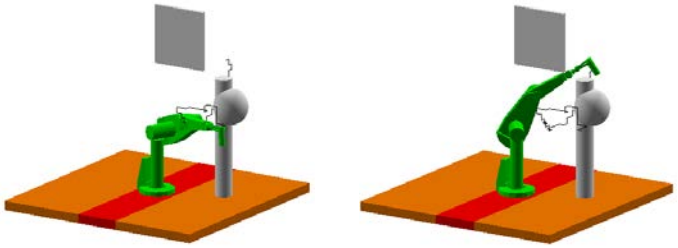


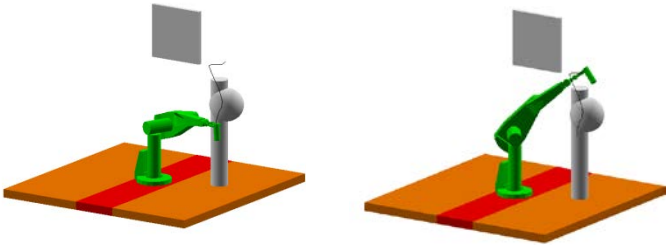
Figure 13 Third obstacle introduced obstructing the path (algorithm G)

The problem is solved again. The new solutions obtained using algorithms G and GA are shown in Figs. 14 and 15.



Initial Configuration Final Configuration

Figure 14 New path obtained using algorithm G



Initial Configuration Final Configuration

Figure 15 New path obtained using algorithm GA

The numerical results obtained for example nº 1 are shown in Tables 3 and 4.

Table 3

Example 1 Fixed-based Robot	Distance Travelled (in m.)				
	Indirect Algorithm: Seq	Simultaneous Algorithm: A*	Simultaneous Alg.: Uniform Cost	Simultaneous Algorithm: Greedy	Genetic Algorithm
With 0 obstacles	3,73	3,55	3,55	3,65	3,1425
With 1 obstacle.	3,72	3,44	3,44	3,55	3,1671
With 2 obstacles	4,50	4,86	4,86	5,12	3,4890
With 3 obstacles	5,42	5,41	3,55	6,83	3,6139

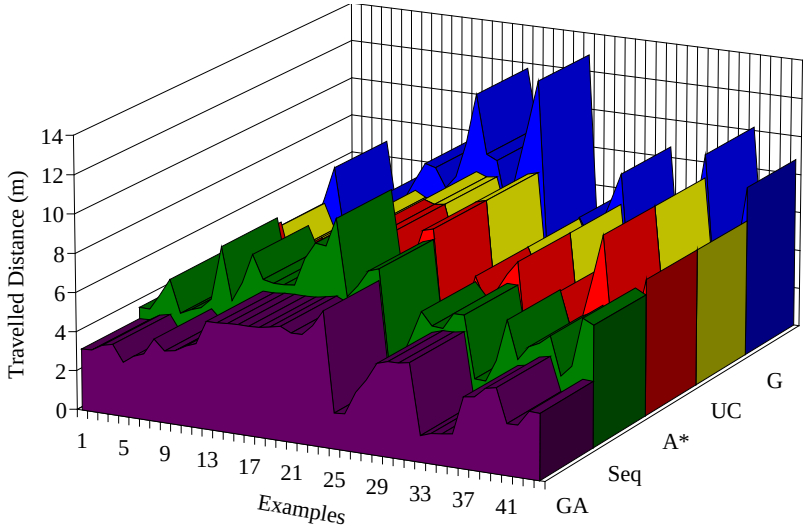
Table 4

Example 1 Fixed-based Robot	Computational Time (in s.)				
	Indirect Algorithm: Seq	Simultaneous Algorithm: A*	Simultaneous Alg.: Uniform Cost	Simultaneous Algorithm: Greedy	Genetic Algorithm
With 0 obstacles	36,19	253,05	773,67	1,20	4442
With 1 obstacle.	8,83	57,44	214,81	0,86	4034
With 2 obstacles	15,27	78,48	237,59	3,04	16166
With 3 obstacles	30,73	110,51	148,75	24,68	6991

6.4 Results obtained for all the examples

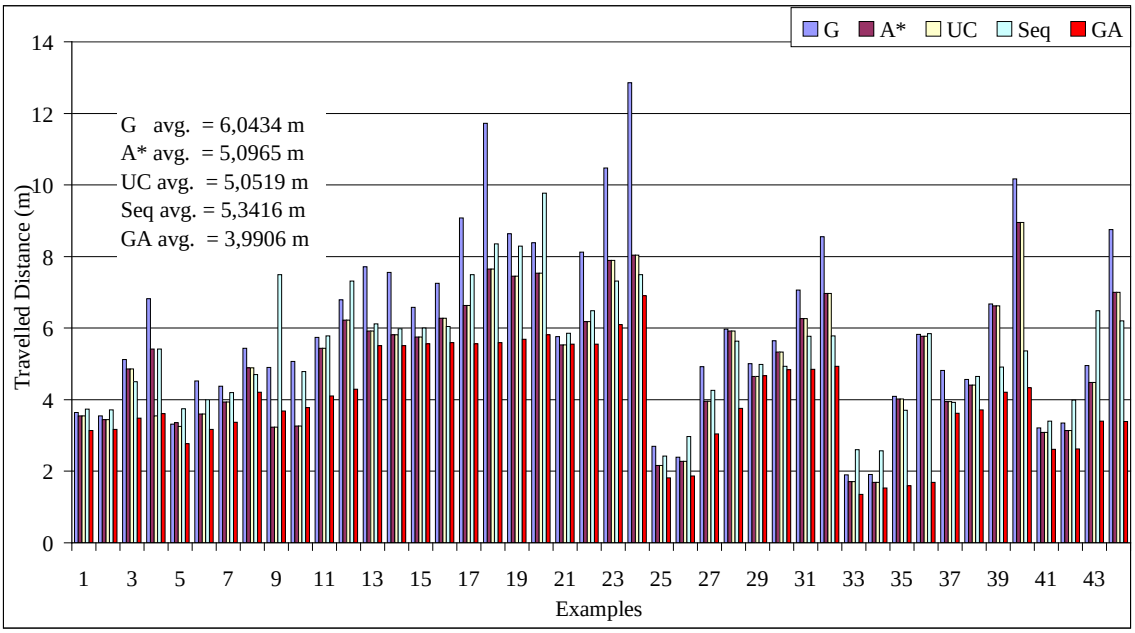
Next, we show the graphs generated for all the examples, computing the computational time required and the distance travelled by the robot.

Graph 1 and Graph 2 show the distance travelled by the robot in two different formats. Note that the genetic algorithm (GA) yields the best results.



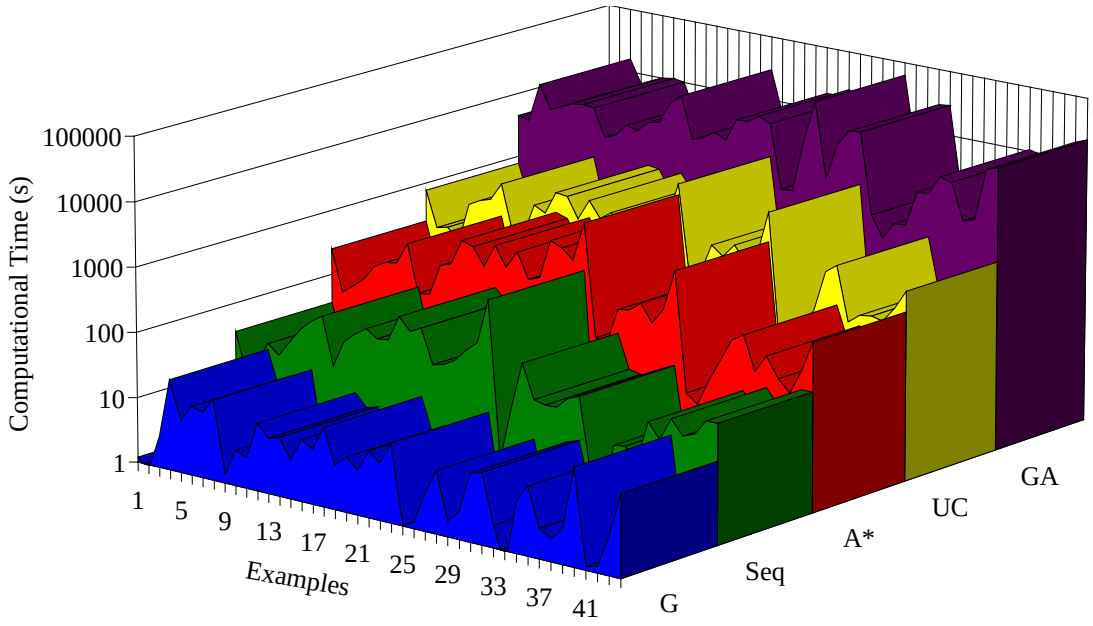
Graph 1 Distance travelled by the robot

It shows that the genetic algorithm search scheme produces the lowest distance travelled. We can see these data from another point of view in Graph 2.



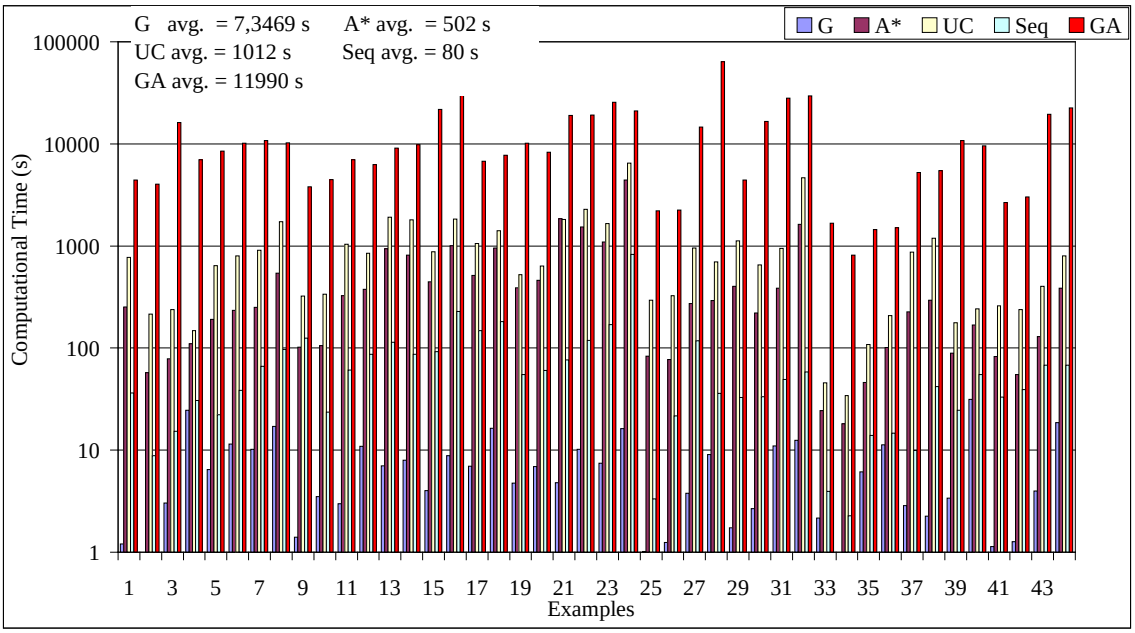
Graph 2 Distance travelled in bar format

Graph 3 and Graph 4 show the computational time required to get a solution in two different formats. Note that the greedy algorithm (G) yields the best results that are the lowest computational times.



Graph 3 Computational time required

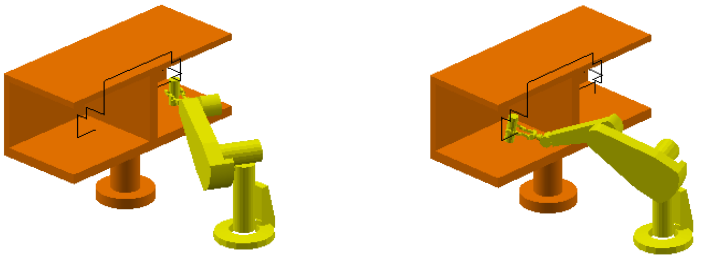
It shows that the greedy algorithm search scheme yields the lowest computational time required to get a solution. We can see the these data from another point of view in the Graph 4



Graph 4 Computational Time required in bars format

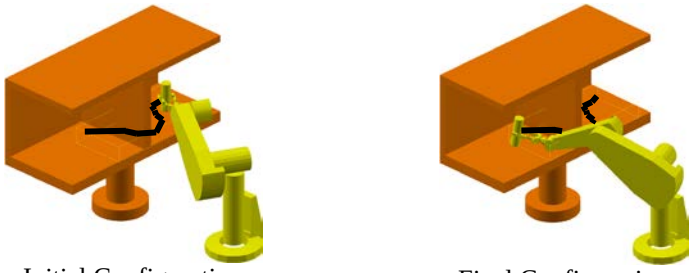
6.5 Industrial Application Example

Next we present an industrial application. We can see the robot in the workspace with different initial and final configurations and the obstacles:



Initial Configuration Final Configuration

Figure 16 Distance travelled by the robot



Initial Configuration Final Configuration

Figure 17 Distance travelled in bar format

Table 5: Configurations to be joined (example nº 2)

Joint Nº	Initial Configuration	Final Configuration
1	-27.57º	24.04º
2	-28.26º	-28.26º
3	148.61	140.78
4	0.5º	0.5º
5	-31.96	-31.96º
6	0.0º	0.0º
7	0.0 mm	0.0 mm

Table 6: Obstacle Location (in m.) (example nº 2)

1 Obst.	Prismatic	$a_1^{PO} = (1.3, -0.6, 1.1)$	$_{11}^{PO} = (0.7, -0.6, 1.1)$	$q_{21}^{PO} = (1.3, 0.9, 1.1)$	$q_{31}^{PO} = (1.3, -0.6, 0.9)$
2 Obst.	Prismatic	$a_2^{PO} = (1.3, -0.6, 0.5)$	$q_{12}^{PO} = (0.7, -0.6, 0.5)$	$q_{22}^{PO} = (1.3, 0.9, 0.5)$	$q_{33}^{PO} = (1.3, -0.6, 0.4)$
3 Obst.	Prismatic	$a_3^{PO} = (1.3, 0.1, 0.5)$	$q_{13}^{PO} = (0.7, 0.1, 0.5)$	$q_{23}^{PO} = (1.3, 0.1, 1.1)$	$q_{33}^{PO} = (1.3, 0.1, 0.4)$

7. Conclusions

In this paper:

1. Authors summarize the characteristics of five algorithms to solve the path planning problem for industrial robots operating in 3D complex environments.
2. A comparison between those five algorithms has been presented to analyze their performance, determining which algorithm gives the best results through the analysis of the working parameters.
3. The first algorithm is an indirect method (called the sequential algorithm), the next three algorithms (A*, Uniform Cost and Greedy) are based on a direct approach and finally the fifth algorithm is based on genetic techniques.
4. These algorithms can be applied to any industrial robotic systems (by changing the significant points which serve to define the robot) and can solve the path planning problem.
5. The application examples have been developed using a PUMA 560 robotic system.

An analysis of the results indicates the following points:

a) The new procedure introduced in this paper (the genetic algorithm) yields the best results from the point of view of the distance travelled. However, the values of the computational time needed to obtain a solution are greater than values obtained using the rest of algorithms. This is not a drawback when the robot programing is off-line.

b) With regard to computational time, it is worthy pointing out the excellent results obtained by the greedy algorithm. However this algorithm doesn´t gives the best results from the point of view of the distance travelled.

Depending on the working parameter (computational time or distance travelled) used to generate the path, the greedy algorithm or the genetic algorithm will be chosen.

References

Brooks, R. A., and Lozano-Pérez, T. (1983), "A Subdivision Algorithm Configuration Space for Findpath With Rotation", *International Joint Conference on Artificial Intelligence*, pp. 799-806.

Brooks, R. A., (1983), "Solving the Find-Path problem by Good representation of free space", *IEEE Transactions on System, Man y Cybernetics*, Vol. 13, no 4, pp. 190-197.

Davidor, Y. (1991), "Genetic Algorithms and Robotics, a heuristic strategy for optimization", *World scientific series and automated systems*, Vol.1.

Dubowsky, S. and Shiller, Z., (1984), "Optimal dynamic trajectories for robotic manipulators", *Fith CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, Udine, Italy, pp 96-103.

Fares, A.D. et al. (2007), "Path planning optimization of industrial robots using genetic algorithm", *Proc. of 16th Int. Workshop on Robotics in Alpe-Adria-Danube region*, Liubliana, pp. 104.

Gupta, K., del Pobil A. P., (1998), "Practical Motion Planning in Robotics: Current Approaches and Future Directions", John Wiley & Sons, West Sussex.

Nilsson, N., Hart, P., and Raphael, B. (1968), "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *EEE Trans. on Systems Science and Cybernetics*, Vol. 4, no. 2, pp. 100-107.

Hsu, D., Kindel, R., Latombe, J.C. & Rock, S., (2002), "Randomized Kinodynamic Motion Planning with Moving Obstacles", *Int. J. of Robotics Research*, Vol. 21, no 3, pp. 233-255 ,

Isto, P., (1996), "Path Planning by Multiheuristic Search via Subgoals", *Proceedings of the 27th International Symposium on Industrial Robots*, CEU, pp. 712-726.

Khatib, O., (1985), "Real-time obstacle avoidance for manipulators and mobile robots", *IEEE International Conference on Robotics and Automation*, St. Louis, pp. 500-505.

Kavraki, L.E., (1994), "Random Networks in Configurations Space for Fast Path Planning", doctoral diss., Dept. of Computer Science , Stanford University, CA.

Kavraki, L. E., Svestka, P., Latombe, J. C. & Overmars, M. H., (1996), "Probabilistic roadmaps for path planning in high dimensional configuration spaces", *IEEE Transactions on Robotics and Automation*, Vol. 12, 566-580.

Kavraki, L. E. and Latombe, J. C., (1998), "Probabilistic Roadmaps for Robot Path Planning". Practical Motion Planning in Robotics: Current Approaches and Future Challenges", John Wiley, pp. 33-53.

Lozano-Pérez, T., (1983), "Spatial Planning: A Configuration Space Approach", *IEEE Transactions on Computers* Vol.32, no. 2, pp. 108-120.

LaValle, S. M. and Hutchinson, S. A., (1996), "Optimal motion planning for multiple robots having independent goals", *Proc. IEEE Int. Conf. Robot. & and Autom*., pp. 2847-2852.

LaValle, S.M., Kuffner J. Jr., (2000), "Rapidly-Exploring Random Trees: Progress and Prospects", *2000 Workshop on the Algorithmic Foundations of Robotics*.

Mata Amela, V., Valero Chuliá, F.J., "Algorithms for Robot Path Planning Among Obstacles", *23 rd International Symposium on Industrial Robots* I.S.B.N.: 84-604-3652-7

Monteiro D. and Madrid, M., (1999), "Planning of robot trajectories with genetic algorithm", *Proceedings of the First Workshop on Robot Motion and Control*, pp. 223-228,

Ó'Dúnlaing, C., and Yap, C.K., (1985), "A retraction method for planning the motion of a disc', *Journal of Algorithms*, Vol. 6, no 1, pp. 104-111

Rubio, F. et al. (2009), "Direct step-by step method for industrial robot path planning". *Industrial Robot*, Vol. 36, no 6, pp. 594-607.

Rubio F., (2006), "Planificacion de trayectorias de robots industriales. Análisis en entornos con obstáculos", Doctoral thesis, Universidad Politécnica de Valencia, España.

Schwartz, J. T. and Sharif, M., (1983), "On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds", *Adv. Appl. Math*.

Valero, F. J. et al. (1997), "A formulation for path planning of manipulators in complex environments by using adjacent configurations", *Advanced Robotics*, vol. 11 pp. 33-56.

Yang, Z. Q., Liu, L.B. and Yang, W.D., (2008), " Flexible Inspection Path Planning Based on Adaptive Genetic Algorithm", *Proceedings of Control and Decision Conference* 2008 (China) , pp. 1558-1563.