Using Genetic Algorithm With Variable-length Individuals For Planning Two Manipulators Motion

J. RIQUELME[‡], M.A. RIDAO[*] , E.F. CAMACHO[*] , M. TORO[‡]
‡ Dpto. Lenguajes y Sistemas Informáticos. Facultad de Informática y Estadística.
* Dpto. Ingeniería de Sistemas y Automática. Escuela Superior de Ingenieros.
Universidad de Sevilla (SPAIN).

Abstract

A method based on genetic algorithms for obtaining coordinated motion plans of manipulator robots is presented. A Decoupled Planning Approach has been used; that is, the problem has been decomposed into two subproblems: path planning and trajectory planning. This paper focuses on the second problem. The generated plans minimize the total motion time of the robots along their paths. The optimization problem is solved by evolutionary algorithms using a variable-length individuals codification and specific genetic operators.

1 Introduction

The problem is to plan a collision-free motion (obstacles and other robots), from an initial configuration to a goal configuration. The most extended approach to this problem is to decompose it into two subproblems: path planning and trajectory planning. Many algorithms to solve this problem can be found in the literature [1,2,3,4].

The solution obtained by most of these algorithms is a robot trajectory. These trajectories are very difficult to implement in most industrial robots, because they require the internal controller of each articulation to be fully available to the user.

A method is presented in [5,6] to minimize the total motion time of the robots along their paths. This method is used in this paper to find a collision-free motion plan for two robots, and evolutionary algorithms with three different chromosome codification are presented to solve the optimization problem. In this paper a genetic algorithm where the length of the individuals is variable [7] is proposed. Also, new genetic operators adapted to this codification are presented.

Supported by DGCYT from the Spanish Ministerio de Educacion under grants TAP-95-0370 and TIC-94-0282

2 Problem Statement

The problem can be stated as: Given two robots R_1 and R_2, a set of known fixed obstacles and the initial and final configurations of R_1 and R_2; find a coordinated motion plan for the robots from their initial configuration to their final configuration avoiding collisions with environments obstacles and themselves The use of a Decoupled Planning approach needs a fixed obstacle collision-free path to be previously obtained for each of the robots. The paths which the robots are expected to follow are assumed to be given as a parametrized curve in the joint space, where λ parameter is the distance along the path. The *Coordination Space* (*CS*) is defined as the R^2 region

$$CS = \left\{ (\lambda^1, \lambda^2) \ / \ 0 \le \lambda^j \le \lambda^j_{\max} \text{ with } 1 \le j \le 2 \right\}$$

Anypath from (0,0) to ($\lambda 1,m_{ax},\lambda 2,m_{ax}$) determines a coordinated execution of the two paths, and is called a *Coordination Path* (CP). The *Collision Region* (CR) is defined as the set of points in CS where a collision between two manipulators is produced. In order to reduce the search space in CS, a discretization of each path has to be made, so the path is divided into several equal intervals. Let's number the intervals of each path from 1 to max_j and the ordered set of intervals is called Ω_j . A cell is defined as the subspace formed by one interval of the paths of each of the robots and is represented as the pair (n_1,n_2). With these discretized paths, CS is transformed into an array of cells, the *Coordination Diagram* (CD). Cell. Let's notate C_0=(0,0) and C_{max}=(max_1,max_2). A cell (n_1,n_2) is considered collision FREE if every point inside the cell does not belong to the collision region. Otherwise, it is considered an OBSTACLE cell.

Robots can be synchronized using *Synchronization Points* (SP), that is, a point in CD, which any CP will necessarily pass through. When the robot arrives at that place on its path, it will stop until the other robot arrives at its respective points. To avoid a collision it is possible to alter the CP defining the number and position of the SP, determining the total motion time.

Let's consider a rectangle formed by free cells in CD and let's consider the motion of the robots from the lower left corner cell to the upper right corner cell. Any trajectory defined for each robot between these two points in CD will always be a collision-free CP. This class of rectangles is going to be called *Free Rectangles*.

Let's consider a set of Free Rectangles, connected in such a way that the upper right corner of one rectangle is the lower left corner of the next. Furthermore, the lower left corner of the first one is the lower left corner of the whole CD, and the upper right corner of the last rectangle is the upper right corner of CD. This set of rectangles is a *Free Rectangle Sequence*, and the intersection points between two rectangles will be the SP. This constraint is very easy to implement using any robot programming language.

The problem can be stated as that of finding a Free Rectangle Sequence that minimizes the total execution time necessary for the robots to complete their whole path. The main variables used to find this sequence are the number of SP and the position of these points in CD. A complete description of the method can be found in [5,6].

3.- The Proposed Evolutionary Algorithms

Three different evolutionary algorithms to solve the optimization problem are presented in this paper. The main differences among them are the codification of the individuals and the genetic operators. While the first one uses the classical binary codification of genetic algorithm, the others use a specific integer codification with new genetic operators. One of them uses fixed-length chromosomes and the other variable-length individuals.

3.1.- Fixed-length integer codification.

In the solution proposed in [5], each individual is formed by a fixed number of points represented by their absolute coordinates in Ω. Different crossover, mutations and specific genetic operators were studied in the same paper, obtaining good results. The main drawback of this codification is the use of fixed chromosome length. In order to avoid this disadvantage, a new variable length codification is proposed in this paper.

3.2.- Variable-length integer codification.

Chromosome representation of the individuals.Each individual is represented by a increasing SP sequence. The chromosomes that represent each solution have variable lengths. If n is the length of a chromosome, then a SP sequence will be determined by n+2 points, where $(x_0,y_0)=(0,0)$ and $(x_{n+1},y_{n+1})=(max_1,max_2)$. An individual is valid if it forms an increasing Sequence of Free Rectangles. Finally, when a genetic operator is applied, it is possible for a non-increasing SP Sequence to be obtained (non-acceptable individual).

Generation of the initial population. The initial population is selected randomly.A uniform distribution of the number of points between 1 and *NMAX* has shown to be inappropriate in the tests to represent the lenght of the individuals. The reason is that non-optimal solutions, with few points, can constitute local minima. For this, an increasing probability distribution from 1 (minimum) to *NMAX* (maximum) was selected.

Once the number of points of an individual n is selected, its values are created as follows: two sets of n random values in [0,1] are generated, then they are ordered in an increasing way and projected on $[0,max_1]$ and $[0,max_2]$ respectively.

Fitness measure The evaluation of the fitness measure will consider valid and non-valid individuals. For valid ones, the fitness function gives the total execution time needed by the robots to complete their paths, when SP are placed in the positions defined by the individual specifications (See [5] for a more detailed description).

The fitness function for non-valid individuals is completely different. The function must measure how far it is from a valid individual. The function considered is $f(N)=K+nco$, where K is an offset, i.e. a high value with respect to the value associated with the valid individuals, and *nco* is the number of obstacle cells inside the rectangle sequence.

Genetic operators.
Crossover Operator: Given two individuals S^1 and S^2 formed by sequences of n and m SP respectively, the

idea is to obtain another SP sequence through genetic information exchange from parents S^1 and S^2. The following method is proposed:

A SP of S^1 is randomly selected, called *P*. Then, *Q*, a SP of S^2 is selected. *Q* is the first SP that has both coordinates *x* and *y* greater than the respective coordinates of *P*, so the resulting child always will be an increasing SP sequence. It no point of S^2 verifies this limitation, then the child is returned as a copy of S^1. If point *Q* exists, a new individual is formed by the first points of S^1 (*P* inclusive), followed by the points of S^2 from *Q* (inclusive) to the end of S^2 (see fig. 1).

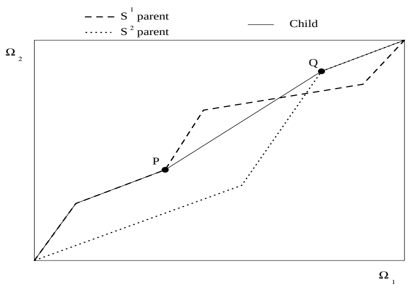

Fig. 1 - Crossover operator

Mutation Operator: . Two groups of mutation operators are proposed in this paper:

- *Slight Operators*, which vary the genetic information sightly. These operators accomplish a local search, that is, in the neighbourhood of a solution.
- *Strong Operators*, which modify the structure of a solution. The objective is to extend the search to new regions, permitting the algorithm to escape from local minima.

Experimental results show that the most important improvement are obtained with the slight mutation. Nevertheless, the strong mutations play a very important role in the process, avoiding the algorithm to remain trapped in a local minima, even when these mutations rarely improve the individuals. The Slight mutations are defined as follows: A SP (x_k,y_k) of individual *S* is selected randomly and also an integer *m* between -*MUTMAX* and *MUTMAX* is chosen too, where *MUTMAX* is the maximum permitted mutation. The mutation consists of substituting the point (x_k,y_k) by (x_k+m,y_k+m), where the new point will be placed in the grey rectangle.

On the other hand, the strong mutations can be defined, for example, in the following way: Two consecutives SP point $P=(x_k,y_k)$ and $P'=(x_{k+1},y_{k+1})$ are chosen, and applies them the slight mutation with a given probability. Thereafter, this mutation adds a new SP between both of them. For this, the differences $d_x=x_{k+1}-x_k$ and $d_y=y_{k+1}-y_k$ are obtained. Then, another two integers are chosen randomly: v_x between 1 and d_x-1 and v_y between 1 and d_y-1. The new point is located in *S* below *P* with coordinates (x_k+v_x,y_k+v_y). (Fig. 2)

Restrictions to the individuals: When these operators are applied, the resulting individual may not verify the condition of being an increasing sequence. To solve this problem, the conflicting coordinates are made equal to the same coordinates of the following point. For example, given two consecutive points $P=(x_k,y_k)$ and $P'=(x_{k+1},y_{k+1})$, if $y_k>y_{k+1}$ then $y_k=y_{k+1}$ is made.

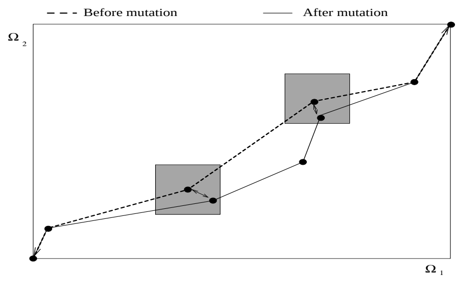

Fig. 2 - Segment mutation

Evolutionary algorithm characteristics. An elitist evolutionary algorithm has been used, where the best individual of each generation is replicated in the following one. A percentage of the offspring is obtained through parents mutations selected with a probability proportional to its fitness. The rest of the offspring is obtained through parents crossover (also selected as a function of its fitness), and after, one of the above defined mutation operators is applied with a random probability.

4 Application Examples

The proposed algorithm has been implemented and applied to several examples in order to study its efficiency. Only one example is presented in this paper. The example corresponds to the motion of two SCORBOT and sixteen collision regions and 180×180 cells (Fig. 3). It is an iterative motion represented in Fig. 4 The motion from Fig. 4-1 to 4-2 and again to the initial configuration 4-3, is repeated twice.

To compare the different evolutionary algorithms, 40 simulations have been executed with a 100 individuals population and 300 generations. Individuals in binary codification and fixed-integer are formed by 20 SP. Also, the maximum number of SP of the initial population is 20 in integer-variable codification, in order to use similar initial information.

The best results are obtained with integer variable codifications. Also, fixed codification reached a lower minimum value, but standard deviation is three times greater.

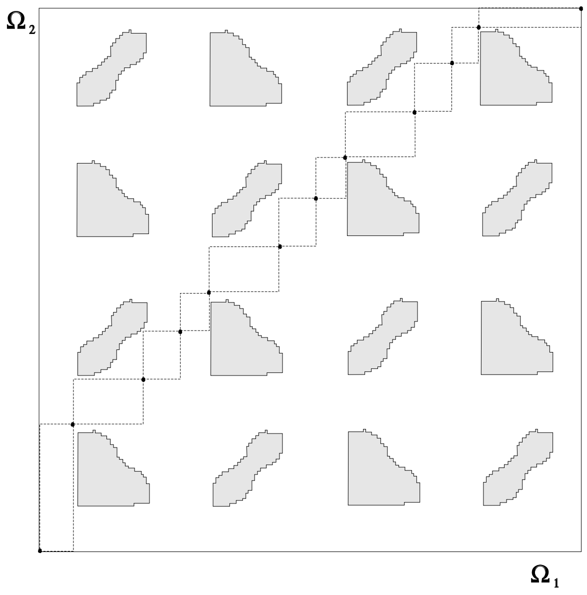

Fig. 3 - Coordination Diagram and the best SP sequence

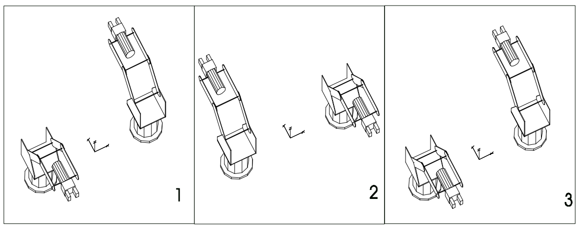

Fig. 4 - Initial position (1), intermediate (2) and goal position (3)

Finally, notice two other advantages in using variable codification. First, its computational cost of crossover and mutation operations may be even 60% of the fixed codification cost, because normally, the number of SP is smaller in variable codification. Also, it is not necessary to previously know an estimation of the number of SP of the optimal solution.

5 Conclusions

This paper describes a method to generate collision free coordinated motion plans in multirobots systems. The method tries to find a SP sequence that minimizes the total execution motion time. This optimization problem has been solved using different evolutionary algorithms. Three chromosome codifications and different genetic operators have been implemented. Better results are obtained in the application examples when using an integer variable codification.

Table I - Results (Time in seconds)

Codification	Avg.	σ	Min.
Binary	54.78	0.56	53.25
Integer fixed	42.23	2.78	37.63
Integer Var.	41.19	1.16	38.63

References

[1] J.C. Latombe, *Robot Motion Planning*, Klewer Academic Publishers, 1991.

[2] K. Kant and S.W. Zucker, *Toward Efficient Trajectory Planning: The Path-Velocity Decomposition.* Int. J. of Robotics Research, 5 (3).1986.

[3] B.H. Lee and C.S.G. Lee, *Collision-Free Motion Planning of Two Robots*, IEEE Trans. on SMC, pp 21-32,Vol 17 no 1, Jan-Feb 1987.

[4] P.A. O´Donnell and T. Lozano-Pérez, *Deadlock-Free and Collision-Free Coordination of Two Robots Manipulators*, Proc. of the IEEE Int. Conf. on Robotics and Automation, pp 484-489, 1989.

[5] M.A. Ridao, *Generación Automática de Trayectorias Libres de Colisiones para Múltiples Robots Manipuladores*. Ph. D. Thesis. Universidad de Sevilla. 1995.

[6] M.A. Ridao, J. Riquelme, E.Camacho and M. Toro, *Coordinated Motion Planning of Manipulators by Evolution Strategies*. Proc. of the X Int. Conf. on Applications of AI in Engineering. Udine.1995.

[7] S.F. Smith, *A Learning System Based on Genetic Adaptive Algorithms.* Ph. D. Thesis. University of Pittsburgh. 1980.