Using genetic algorithms for robot motion planning

Juan Manuel Ahuactzin(i)*, **El-Ghazali Talbi**(ii) **,
Pierre Bessiere(ii), **Emmanuel Mazer**(i) ***

Institut National Polytechnique de Grenoble
46, Avenue Félix Viallet, 38031 Grenoble cedex - France
Tel: (33) 76.57.48.13 Fax: (33) 76.57.46.02

Abstract. We present an ongoing research work on robot motion planning using genetic algorithms. Our goal is to use this technique to build fast motion planners for robot with six or more degree of freedom. After a short review of the existing methods, we will introduce the genetic algorithms by showing how they can be used to solve the invers kinematic problem. In the second part of the paper, we show that the path planning problem can be expressed as an optimization problem and thus solved with a genetic algorithm. We illustrate the approach by building a path planner for a planar arm with two degree of freedom, then we demonstrate the validity of the method by planning paths for an holonomic mobile robot. Finally we describe an implementation of the selected genetic algorithm on a massively parallel machine and show that fast planning response is made possible by using this approach. Keywords: robot motion planning, genetic algorithms, parallel algorithms.

1 Introduction

Today most of the robot motion planners are used offline: the planner is invoked with a model of the environment, it produces a path which is passed to the robot controller which, in turn, execute it. In general, the time necessary to achieve this loop is not short enough to allow the robot to move in a dynamical environment. Our goal is to try to reduce this time to be able to move a six degree of freedom arm among moving obstacles. In order to achieve this goal we have chosen genetic algorithms for the following reasons:

- They are well adapted to search for solutions in high dimensionality search space.
- They are very tolerant to the form of the function to optimize, for instance these functions do not need to be neither differentiable or continuous.
- They can easily be implemented on a massively parallel machine and they achieve super-linear speed up with the number of processors [8].

In addition we have tried to take advantage of tools already existing in classical algorithms used in path planning, such as range computation, slice representation of the configuration space and on demand computation of the configuration space.

2 Previous work

Designing a new path planner is a classical exercise in robotic research and it remains a very active field in robotics. A review of the existing approaches can be found in Latombe's book [1]. Recently a real time path planner system has been demonstrated by [2]. The method works for a three degree of freedom robot assuming the three dimensional configuration space has been precomputed. The system described in [3] is probably one of the fastest system if one consider the time necessary to compute the configuration space and the time necessary to search through it. It uses an efficient way of computing the configuration space and it is implemented on a Connection Machine.

3 Genetic algorithms

Genetic algorithms are stochastic search techniques, introduced by Holland [4] twenty years ago, they are inspired by adaptation in evolving natural systems. Development of massively parallel architectures made them very popular in the very last years. To introduce the genetic algorithms and their application in path planning, we will first solve a simple robotic problem: the invers kinematic problem.

---

*(i) Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle Grenoble, France.
**(ii) Laboratoire de Genie Informatique, IMAG, Grenoble, France.
***This work has been made possible by: Le Centre National de la Recherche Scientifique, Consejo Nacional de Ciencia y Tecnologia (Mexico) and ESPRIT "Supernode 2" project.

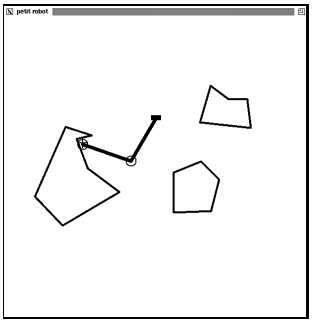

Figure 1: A planar arm with two degrees of freedom.

3.1 Solving the invers kinematic problem

Lets consider an arm with two degree of freedom an lets denote $\theta = (\theta_1, \theta_2)$ a particular configuration of the arm. One version of the invers kinematic problem can be stated as a minimization problem:

$$\min_{\theta} f(\theta) \; with \; f(\theta) = \begin{cases} \|d(\theta) - X\| & \text{if } \theta \notin \text{C-obstacles} \\ +\infty & \text{otherwise} \end{cases}$$

Where $d(\theta)$ denotes the direct kinematic function and X the desired cartesian location for the extremity of the arm.

In general, it is very hard to get an analytical expression of f given a set of obstacles, but computing its value for a given θ can be done easily.

In the rest of this section we show how to use GA to solve the invers kinematic problem with obstacles.

1. **Coding the problem.** The search space (here $[0, 2\pi] \times [0, 2\pi]$) is discretized: In our case a configuration is represented by two integers (i_1, i_2), where i_1 and i_2 belong to $0, \ldots, 255$. We denote as b the string of bits resulting from the concatenation of the binary representation of i_1 and i_2, for example if $i_1 = 1$ and $i_2 = 2$: $b = 00000010000010$. b codes a point of the discretized search space. In the GA terminology b is referred as a "individual".

2. **Generating an initial population.** A random set of n "individuals" $B = \{b_i\}_{i=1,n}$ is generated. For example figure 1 represents our small robot in a clustered environment and figure 2 the associated configuration space. The small $\bullet$ indicates a particular $b_j \in B$, the $\circ$ sign indicates the desired locations to place the extremity of the arm in the desired cartesian position.

3. **Operating a selection.** The function f is applied to each member of the population and the "individuals" are ranked accordingly to their associated value of f. After this step, the closer a configuration brings the extremity of the arm to the desired goal without collision, the better its rank will be.

4. **Creating couples and combining individuals.** A set of n couples $\{(b_i, b_j)\}, b_i, b_j \in B$ is generated. For each couple, b_i and b_j are obtained by randomly picking an element of B with a probability proportional to their rank.

   For each couple the "cross-over" operation is used to produce two new individuals nb_1 and nb_2. This operation consists of generating at random a place to "cut" the binary strings which codes the two configurations defining the couple, to "glue" the left part of b_i with the right part of b_j to obtain nb_1 and to "glue" the right part of b_i with the left part of b_j to obtain nb_2. The new individual of the population is the individual of the set $\{b_i, b_j, nb_i, nb_j\}$ which minimize f. After this step a new population is generated. It can be shown [5] that the average value of f is lower than the previous population.

5. **Generating "mutants" (optional)** One strategy to escape from local minima is to introduce noise in the algorithm. This can be done by flipping at a given bit a given individual of the population. The individual and the bit bit are chosen at random.

6. **Termination conditions** There are two termination conditions:

   (a) The absolute minima is obtained for one element of the population and a solution is found.

   (b) The population stabilized and the algorithm is stuck in a local minimum.

   If neither conditions is true the step # 3 is applied to the new population.

In our example the population converge quickly towards the two solutions (see figure 3). It can be shown [5],[6] that the GA work by interratively selecting an increasingly smaller subspace of the configuration space with a good average value for the fitness function.

4 A simple path planner

In this section we describe a simple path planner for a planar arm with two degrees of freedom. By restricting ourself to two dimensions we can graphically represent the configuration space and give the reader a better feeling of the method. However the proposed method does not make any hypothesis about the number of degree of freedom and can be used without modification for arms with having a larger number of degree of freedom.

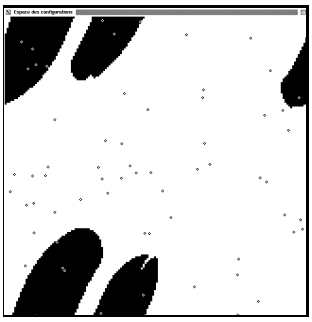

Figure 2: Initial distribution of the population in the configuration space .

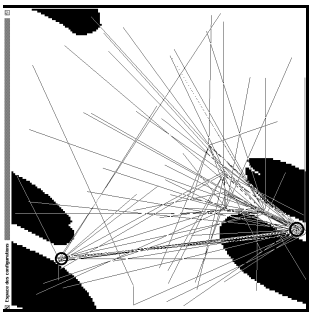

Figure 3: The population converges twoards the two solutions of the invers kinematic problem.

To plan a path with a GA we use a quite unexpected search space. Instead of searching for path in the configuration space, we consider a discretized subset of all the possible paths (with or without collision) starting from the initial configuration. For example, in the case of our planar robot we consider the following subset of possible motions: "move from θ_1^0, move from θ_2^1, move from θ_1^2, move from θ_2^3, ... move from θ_1^i, move from θ_2^{i+1}, ..., θ_1^{2k}, move from θ_2^{2k+1}" where each θ_i^j is discretized on q bits.

We call such a motion a Manhattan motion of length k and we denote the set of all these paths as $M(k,q)$.

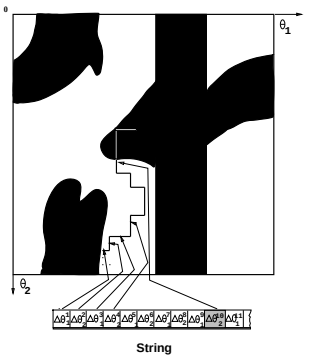

Figure 4: A Manhatan motion in the configuration space.

Figure 4 shows such a motion in the configuration space.

They are four main reasons to consider Manhattan motions:

1. They define a naturally redundant search space which is well suited for stochastic methods.
2. They are easy to test: When a Manhattan motion is executed only one link move at a time, and thus the test can be done by successively computing the legal ranges of motion for a single degree of freedom [7] . Note that this last property remains true for three dimensional objects and for robot with many degree of freedom.
3. Each time a Manhattan motion is tested it is possible to fill a part of the configuration space if it is represented with slices(see [7]). Note that this representation of the configuration space can be use to speed up further evaluations of new Manhattan motions. This technique permits the lazy evaluation of the configuration space while trying to reach the goal.
4. They can be easily stretched([2])

We can now define the two ingredients necessary for our GA:

Coding the element of the search space Let $P \in M(k,q)$ we can code P with $k \times q$ bits (in practice k and q are experimentally chosen): $\theta_1^0, \theta_2^1, \theta_1^2, \theta_2^3, \ldots, \theta_1^i, \theta_2^{i+1}, \ldots, \theta_1^{2k}, \theta_2^{2k+1}$

Defining the fitness function The fitness function make use of a special accessibility predicate MP defined as follow.

Let I and G two points of the configuration space:

$MP(I,G)$ is true if and only it is possible to find a simple collision free path from I to G (ie: if it is possible to move freely from (θ_1^I, θ_2^I) to (θ_1^G, θ_2^I) and from (θ_1^G, θ_2^I) to (θ_1^G, θ_2^G).

Note that the definition of this predicate can easily be evaluated and generalized. Now, we can define how to compute the fitness function:

Let G be the goal and S_i be the extremity of the i^{th} segment of the Manhattan path P:

```
begin
for i from 0 to 2k
    if MP(S_i, G) return 0
    else if ¬ MP(S_i, S_{i+1}) return ||S_i − G||
         endif
    endif
endfor
return ||S_{2k+1} − G||
end
```

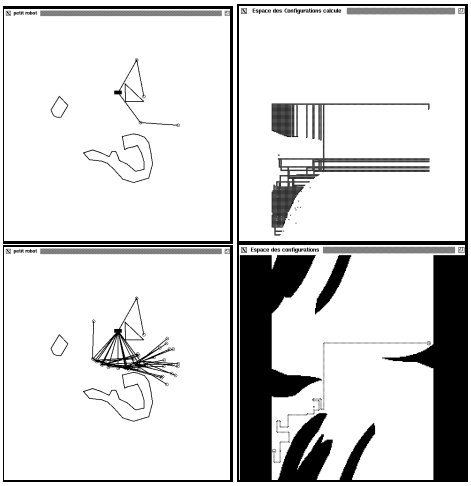

Figure 5: A motion planning problem and the solution in the operational and configuration spaces.

This function simply scan the intermediary points of the candidate path to find if a "direct" move towards the goal, it stops when an obstacle is found and return the distance from the extremity of the last free segment to the goal.

Figure 5 shows the initial and the final position of the robot in the operational space. At the botton of the figure we show a path found with this method in the two spaces. The right top of the figure is the portion of the configuration space which has been evaluated during the planning process, in this particular case only 1/10 of the total configuration space has been evaluated.

5 Planning a path for an holonomic mobile robot

In this section we describe the application of our algorithm to the problem of planning a collision free path for an holonomic mobile robot. The search space is still a subspace of all the possible paths starting from the origin, however the paths are directly coded as a list of "rotate " and "move" commands. The fitness function is also slightly different but uses the same principle of "visibility to the goal " used in the previous case. The Figure 6 shows two successful paths planned with the method. This planner has been implemented on the architecture described bellow. On this architecture (128 Transputers) the planning time was under one second. Since this planner does not make use of a precomputed representation of the configuration space it may be suitable to plan paths in a dynamical environments.

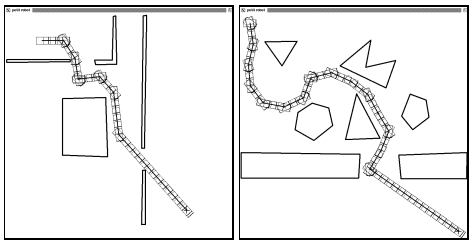

Figure 6: Examples for the holonomic robot.

6 Parallel Genetic Algorithm

Besides the "intrinsic parallelism" of genetic Algorithms [4][5], we designed and implemented a parallel genetic algorithm which shows a remarkable superlinear speed-up in the number of processors [8]. The principle of this algorithm is to distribute the individuals of the population on a set of processors, limiting the reproduction (step 3 and 5 of the standard algorithm) to the nearby individuals. The implementation has been done on a SuperNode [8]c.

The SuperNode is a loosely coupled and highly parallel machine based on transputers. One of its most important characteristics is its ability to dynamically reconfigure the network topology by using a programmable VLSI switch device. This architecture offers a range of 16 to 1024 processors, delivering from

24 to 1500 Mflops performance. The adopted configuration of the machine is a torus. Given the four links of the transputer, each individual has four neighbors (see figure 7). The distribution of the population on the set of processors has been done on a one individual by processor basis. Consequently, at each generation on individual is susceptible reproduce only with its four direct neighbors. The parallel genetic algorithm is as follow:

- **Step 1** Distribute and generate the initial population on the set of processors (one individual by processor).
- **Step 2** Evaluate the initial population **IN PARALLEL**.
- **Step 3** Until the stop conditions are reached do **IN PARALLEL**:
  - Exchange with the four neighbors the individuals (chain of bits) .
  - Do the four possible individual combinations (Cross over), between the local individual and each of the four just acquired neighboring individuals.
  - Do (optionally) the mutation.
  - Select the best of the generated offspring and replace the current local individual.

Note that, unlike for the sequential Genetic Algorithm, we are not able to get the best solution in the network. The communication involved in determining such a solution would be considerable. we only pick up the solution routing through a "spy process" placed at the root processor (see figure 7).

This parallel genetic algorithm and its associated implementation have several important qualities:

- The needed configuration of the network of processors (a torus) is very simple and easy to obtain.
- Communication between processors are only local (limited to the four neighbors) and thus supposes only a very simple message routing capability.
- Due to the strictly local form of the message passing strategy, the amount of communications grows linearly with the number of processors. This proves that huge configuration counting hundreds of processors will still permit a valid implementation of the proposed algorithm.

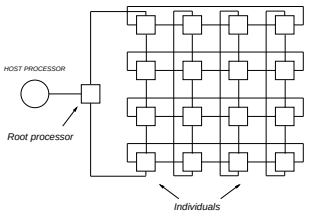

Figure 7: A torus of 16 processors

7 Future work

We plan to integrate this planner in a larger robotic experimentation tested. Our goal is to use real CAD models to evaluate the fitness function. The CAD models of the arms and of the obstacles will be obtained from the ACT [9] simulation package. The range computation used to evaluate the fitness function will use the same algorithm found in the global path planner of ACT. Our goal is to be able to execute the computed trajectory via KALI [10] and to use the second arm as a dynamical obstacle.

References

[1] Jean-Claude Latombe: *Robot Motion Planning*, Ed. Kluwer Academic Publisher, 1991.

[2] Sean Quinlan, Oussama Khatib: *Towards Real-Time Execution of Motion Tasks*, Second International Symposium on Experimental Robotics, Toulouse, June 1991.

[3] Tomás Lozano-Pérez, Patrick A. O'Donnell: *Parallel Robot Motion Planning*, IEEE Int. Conf. on Robotics and Automation, Sacramento, April 1991.

[4] J.H. Holland: *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Pres, 1975.

[5] David E. Goldberg: *Genetic algorithms in search, optimization and machine learning*, The University of Alabama, Addison-Wesley publishing company, inc. 1989.

[6] Davis E. Goldberg: *Genetic algorithms and machine learning*, University of Alabama, Tuscaloosa, University of Michigan, Ann Arbor, in Machine Learning, Kluwer Academic Publishers, 1988.

[7] T. Lozano-Pérez: *A simple motion-planning algorithm for general robot manipulators.*, IEEE Int. Jour. on Robotics and Automation, RA-3, June 1987.

[8] E.G. Talbi, P. Bessiere: *A parallel Genetic Algorithm for the graph partioning problem*, ACM International Conference on Supercomputing, Cologne, June 1991.

[9] Emmanuel Mazer and al. : *ACT a robot programming environment*, IEEE Int. Conf. on Robotics and Automation, Sacramento, April 1991.

[10] V. Hayward, L. Daneshmend, S. Hayati. *An overview of KALI: A system to program and control cooperative manipulators*, Fourth international conference on advanced robotics, 1988.