Genetic Algorithm for Dynamic Path Planning

Ahmed Elshamli, Hussein A. Abdullah, Shawki Areibi
School of Engineering, University of Guelph, CANADA
aelshaml@uoguelph.ca, habdulla@uoguelph.ca, sareibi@uoguelph.ca

Abstract

Optimization in dynamically changing environments is a hard problem. Path planning for mobile robots is a complex problem that not only guarantees a collision-free with minimum traveling distance but also requires smoothness and clearances. This paper presents a genetic algorithm approach for solving the path planning problem in stochastic mobile robot environments. The Genetic Algorithm Planner (GAP) is based on a variable length representation, where different evolutionary operators are applied. A generic fitness function is used to combine all the objectives of the problem. In order to make the algorithm suitable for both static and dynamic environments, problem specific domain knowledge is used.

Keywords: *Genetic algorithm, Path planning, Mobile robot, Dynamic environments.*

1. Introduction

The path planning problem is an optimization problem that involves computing a collision free path between two locations. This type of path planning is used in a large number of robotics applications such as manufacturing, assembly, transportation and services.

This work is built on top of the Evolutionary Planner/Navigator (EP/N) technique [1]. A floating-point variable length chromosome is used, which enables the GA to search the entire solution space. A new approach is designed to deal with uncertainty that ensures population diversity during the search process, which makes the algorithm capable of continuously adapting the solution to changing environments.

The problem background is covered in section 2. In section 3, the proposed GAP is presented. Section 4 presents the conducted experimental results. Conclusions are presented in section 5.

2. Background

The path planning problem is an ordering problem, where a sequence of configurations is sought, starting from an initial location and ending at the goal location. A robot searches for an optimal or near optimal path with respect to the problem objectives.

The path planning problem criteria may include distance, time, and energy. The distance is the most common criterion. However common path planning approaches do not take into consideration path safety/ smoothness. Safety constrains are important to both the robot and its surrounding objects. Smoothness is also another important constraint. Most mobile robots need to consider this constraint because of the bounded turning radius. For example car like robots have this constraint due to the mechanical limitations on its steering angle.

Path planning algorithms are classified according to completeness as *exact* and *heuristic* [2]. Exact algorithms aim to find an optimal solution if one exists, or prove that there is no feasible solution. On the other hand, heuristic algorithms aim to search for a good quality solution in a short time. Exact algorithms are usually computationally expensive, however heuristic algorithms may fail to find a good solution for difficult problems.

Different conventional approaches have been developed to solve path planning problems [3] such as cell decomposition, road map and potential field. Most of these approaches are based on the configuration space concept [4]. In addition to their lack of adaptively and robustness, conventional approaches are not suitable for dynamic environments because they utilize a sequential search algorithm to generate a single solution. This solution may become infeasible when a change in the environment is detected and a new solution has to be generated from scratch. To overcome the weakness of these approaches, researchers have been trying to apply other techniques to solve this problem. Genetic Algorithms have been recognized as one of the most robust search techniques for complex and ill-behaved objective functions [5]. The basic characteristic that makes the GA's attractive in developing near-optimal solutions is that they are inherently parallel search techniques. In the last decade GA have been applied in the field of path planning (see [6], [7] and [8]). However most GA approaches operate in a grid map or utilize a fixed resolution in the search space. When dealing with

CCECE 2004- CCGEI 2004, Niagara Falls, May/mai 2004
0-7803-8253-6/04/$17.00 ©2004 IEEE

dynamic environment most GA approaches do not control the population diversity [5] (which is the main problem with standard genetic algorithms) due to premature convergence. Trojanowski at el. [9] approach to overcome this problem by adding a local memory to each chromosome in the population for the EP/N. This approach is computationally expensive and the performance depends on the environment.

3. GA Planner

3.1 Representation and initial population

In order to allow the algorithm to operate on the entire working space, vertex graphs are used to represent obstacles in the robot environment. Each obstacle is represented by an ordered list of vertices with no restrictions on the shapes or the sizes of the obstacles.

A chromosome represents a path as a sequence of nodes, where each node contains an "x" and "y" coordinates of a point. The first node is the starting point (or the robot current location) and the last node represents the destination point. The number of knot nodes (intermediate nodes) in the path is variable. Figure 1 shows the linked list data structure used to accommodate the variable length path.

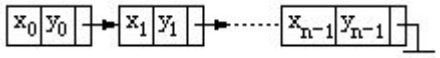


Figure 1: Chromosome data structure

The initial population is generated randomly, where each path has a random number of knot nodes. The only restriction on these randomly generated nodes is that they do not intersect with obstacles (i.e. feasible nodes).

3.2 Path evaluation

Many traditional approaches to path planning consider only the length of the path to compute costs [7]. This evaluation has many drawbacks, since shorter paths may not be safe and smooth and therefore do not represent optimal paths. The same evaluation methodology introduced by Xing at el. [1] is utilized here with some modifications.

A feasible path is evaluated according to the length, smoothness and clearance. A linear combination of these factors is illustrated with the following equation:

$$eval(p) = w_d .dist(p) + w_s .smooth(p) + w_c .clear(p)$$

where w_d, w_s, and w_c represent the weights on the total cost and dist(p), smooth(p) and clear(p) are defined as follows:

- $dist(p) = \sum_{i=1}^{n-1} d(s_i)$, where $d(s_i)$ is the distance between two adjacent nodes.
- $smooth(p) = \sum_{i=2}^{n-1} e^{a(\theta_i - \alpha)}$, where θ_i is the angle between the extension of the two line segments connecting the i[th] knot point. α is the desired steering angle and " a " is a coefficient.
- $clear(p) = \sum_{i=1}^{n-2} e^{a(g_i - \tau)}$, where g_i is the smallest distance from the i[th] segment to all obstacles, and τ is the desired clearance distance.

An infeasible path is evaluated according to the number of intersections with obstacles and the ratio between the numbers of feasible segments and infeasible segments. A penalty function is used to make the worst feasible path better than any infeasible path.

3.3 Reproduction and Genetic operators

In this approach a tournament selection method is used as a selection strategy (selection determines the pair of individuals chosen for recombination). Two paths are selected randomly and the fitter path will be selected as the first parent. The same process is repeated to select the second parent. After the selection process, five operators are used to evolve the selected paths. Each operator application is controlled by its probability. These operators are:

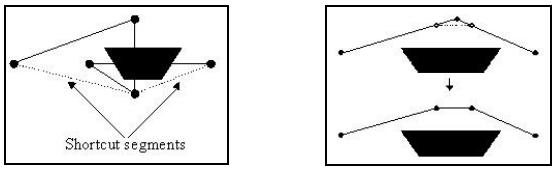
Crossover: This operator combines two selected paths (parents) to generate two offspring's as follows: a random mating knot node is selected on each parent. This node split the path into two parts. The first offspring is generated by combining the first part of the first parent with the second part of the second parent, and the second offspring is generated by combining the first part of the second parent with the second part of the first parent.

Mutation: This operator changes the node coordinates in a path. The mutation window (the range of the coordinates change) decreases as the feasibility ratio (the percentage of the feasible paths in the population) increases.

Repair: This operator is applied to infeasible line segments. Random point(s) around the intersecting obstacle are generated and the operator connects these points to pull the segment around the obstacle.

Shortcut: This operator deletes the intermediate node(s) between two nodes if the segment connects these two nodes is feasible as illustrated in Figure 2a.

Smooth: This operator is applied to feasible paths. A path node is selected and a new node is inserted on each segment such that the segment connecting the new inserted nodes is feasible and the selected node is deleted as illustrated in Figure 2b.



(a) shortcut operator (b) smooth operator

Figure 2: Shortcut and smooth operators.

After applying the operators, the generated offspring's replace their parents. The algorithm is terminated after a fixed number of generations or based on the degree of convergence. Figure 4 shows an overview of the GAP

```
Generate initial population
Evaluate population
While termination condition not met
      Perform evolutionary operators
      Check for new obstacles
      Evaluate population
End while
```

Figure 4: Overview of the GAP.

3.4 Dynamic Planner

This work investigates different strategies that make the algorithm continuously follow the landscape peaks. As reported in [10], the GA should either take an action whenever a change in the environment is detected or avoid convergence and preserve the population diversity all the time. These two strategies have been utilized and the following approaches have been investigated:

Low convergence (LC): In this approach the algorithm runs with low convergence rates, where the mutation probability is set at higher value (50%) and the crossover probability is set at low value (50%).

Random immigrants (RI): Replace a percentage of the population by randomly generated individuals [11]. This operation is repeated at a fixed rate.

Memory (M): The GA stores the best solutions so far and keeps replacing these solutions with the worst individual in the population at a prefixed rate.

Memory and random immigrants (MRI): This approach combines the previous two approaches, the random immigrant rate is set to high and memory replacement rate is set to low.

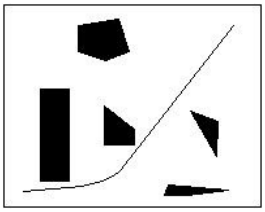
4. Results

The GAP has been implemented in the C programming language and compiled under Solaris/Windows operating systems.

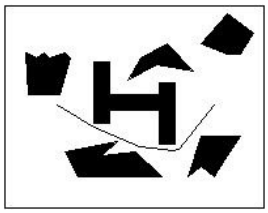
The experiments were conducted for planning paths in 2D environments. Initial experiments revealed that a probability of 0.95, 0.1, 0.1, 0.1 and 0.1 for crossover, mutation, repair, shortcut and smooth respectively work well with different environments.

The performance of the different approaches to adapt to the change in the environment is measured as follows: the GAP was applied to three different environments. These benchmarks are named B1, B2 and B3 (the benchmarks are selected to represent different cases of complexity). First the GAP solves each environment in a standard fashion and no dynamic obstacles were introduced in these runs. Figure 5 shows the near-optimal paths for these benchmarks.

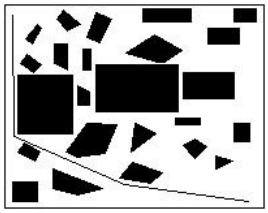
For each environment a new obstacle(s) is introduced during the evolution process such that the introduced obstacles affect the previously found solutions and change the landscape peaks. Each dynamic obstacle is associated with an activation generation number such that when the algorithm reaches the obstacle activation generation number this obstacle is added to the environment.



B1



B2



B3

Figure 5: Different tasks on static environments.

The performance evaluation for each approach is determined by computing the best, worst, average and the standard deviation of 10 runs for each environment. All runs are terminated after a prefixed number of generations. In Table 1 the best performance achieved in each category is shown for each benchmark. Figure 6 illustrates the best obtained solutions for the dynamic environments.

	B1	B2	B3
Best solution	MRI	MRI	MRI
Best Worst	M	LC	LC
Best Average	LC	LC	LC
Best Std.	M	LC	LC

Table 1: Best preformed approaches.

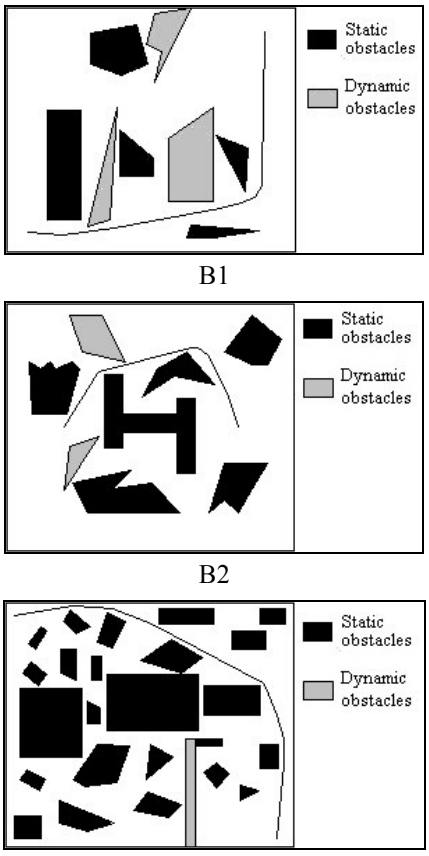


Figure 6: Best solutions.

Results clearly indicate that combing memory with random immigrants generates a near optimal solution. The interesting part of these preliminary results is the effect of the low convergence (LC) approach in providing the most stable performance fro these different tasks.

5. Conclusions

This paper introduced a genetic algorithm approach for solving mobile robot path planning problems in static and dynamic environments. The Genetic Algorithm Planner (GAP) utilizes variable length chromosomes for path encoding. The paths are evolved using specially designed random operators and specific domain knowledge operators. Different approaches that make the algorithm suitable for both static and dynamic environments have been tested.

Preliminary experimental results show that the proposed algorithm is effective and efficient in solving different types of tasks in dynamic environments.

References

[1] J. Xiao, Z. Michalewicz, L. Zhang and K. Trojanowski, "Adaptive Evolutionary planner/navigator for mobile robots," *IEEE Transactions on Evolutionary Computation*, Vol. 1, pp. 18-28, 1997.

[2] Y. K. Hwang and N. Ahuja, "Gross Motion Planning - A Survey," ACM Computing Survey, Vol.24, no. 3, pp. 219-291, 1992.

[3] J. C. Latombe, "Robot motion planning," Kulwer Academic publishers, Boston, MA, 1991.

[4] T. Lozano-Perez and M. Wesley, "An algorithm for planning collision-Free paths among polyhedral obstacles," Communications of the ACM, vol.22, pp.560-570, 1979.

[5] D. E. Goldberg, "Genetic algorithms in search, optimization and machine learning." Addison-Wesley publishing company, 1989.

[6] T. Shibata, T. Fukuda, K. Kosuge, and F. Arai, "Selfish and Coordinative Planning for Multiple Mobile Robots by Genetic Algorithm," Proc. of the 31st Conf. on Decision and Control, Vol. 3, pp. 2686-2691, 1992.

[7] Gerke, M.,"Genetic path planning for mobile robots," Proceedings of The American Control Conference, Vol. 4, pp. 2424 -2429, 1999.

[8] C.H. Leung and A.M.S. Zalzala, "A genetic solution for the motion of wheeled robotic systems in dynamic environments." Int. Conf. on Control '94, Vol.1, pp.760-764,1994.

[9] K. Trojanowski, Z. Michalewicz and J. Xiao, "Adding Memory to the Evolutionary Planner/Navigator," Proc. of the 4th IEEE ICEC, pp. 483-487, 1997.

[10] J. Branke, "Evolutionary approaches to dynamic optimization problems - introduction and recent trends" GECCO workshop on evolutionary algorithms for dynamic optimization problems, pp.2-4, 2003.

[11] J. Grefenstette, "Genetic algorithms for changing environments," In Proceedings of Parallel Problem Solving from Nature (PPSN-2), pp. 137-144, 1992.