International Work-Conference on Artificial and Natural Neural Networks, IWANN'99, Alicante, Spain, 1999.

MOBILE ROBOT PATH PLANNING USING GENETIC ALGORITHMS

Carlos E. Thomaz [†]
Marco Aurélio C. Pacheco [†‡]
Marley Maria B.R. Vellasco [†‡]

[†] Departamento de Engenharia Elétrica
Pontifícia Universidade Católica - PUC/Rio, Brazil

[‡] Departamento de Engenharia de Sistemas e Computação
Universidade do Estado do Rio de Janeiro - UERJ, Brazil

Abstract

Genetic Algorithms (GAs) have demonstrated to be effective procedures for solving multi-criterion optimization problems. These algorithms mimic models of natural evolution and have the ability to adaptively search large spaces in near-optimal ways. One direct application of this intelligent technique is in the area of evolutionary robotics, where GAs are typically used for designing behavioral controllers for robots and autonomous agents. In this paper we describe a new GA path-planning approach that proposes the evolution of a chromosome attitudes structure to control a simulated mobile robot, called Khepera*. These attitudes define the basic robot actions to reach a goal location, performing straight motion and avoiding obstacles.The GA fitness function, employed to teach robot's movements, was engineered to achieve this type of behavior in spite of any changes in Khepera's goals and environment. The results obtained demonstrate the controller's adaptability, displaying near-optimal paths in different configurations of the environment.

Index Terms – Genetic Algorithm, robot, path planning, chromosome.

1. Introduction

The mobile robot path-planning problem is typically formulated as follows: given a robot and a description of an environment, plan a path between two specified locations which is collision-free and satisfies certain optimization criteria [14]. Although a large amount of work exists in this area, conventional approaches tend to be inflexible in rapid responding to changes in the robot's goals and in the environments [14].

One of the main difficulties in designing a control system for autonomous robots is the fact that they interact with an external environment which is not known in advance. In fact, the way robots behave in the environment at a certain time determines the stimuli they will receive as input in the next step. Each robot action has, basically, two different effects : (1) it determines how well the robot's controller performs with respect to the given task; (2) it determines the next input stimuli which will be perceived by the controller. Therefore, defining the correct robot action that the controller shall perform in order to receive good stimuli is extremely difficult because any action may have long term consequences [13].

Since Genetic Algorithms (GAs) are powerful procedures for searching large, complex and multi-modal spaces [6,9], a number of researchers have employed them to solve autonomous agent path planning problems. GA mimics models of natural evolution, having all the benefits of a population based search method. Therefore, it avoids local optima [3] and gradually evolves the control system of an autonomous agent by exploiting the variations in the interactions between the environment and the agent itself. Thus, it appears reasonable to use GA to evolve a control system for a mobile robot. However, the choice of what to evolve is controversial [13]. Brooks [5] proposed to use an extension of Koza's genetic programming technique [11], evolving explicit programs. Hoffmann [10] described a messy genetic algorithm for the automatic design of fuzzy logic controllers. Dorigo and Schnepf [7] proposed to use a form of classifier system. Others proposed to evolve neural networks [1,4,8]. More recently, Jing Xiao and Lixin Zhang [14] combined the concept of evolutionary computation with problem-specific chromosome path structures and operators.

In this work, we present a new approach to this path-planning problem describing a GA controller system that evolves chromosome attitudes structures. These attitudes define the basic motor speed actions and have fitness function engineered to teach a simulated mobile robot, called Khepera*, to reach the goal location, performing straight motion and avoiding obstacles. This learning process does not depend on the obstacle environment distribution, nor on the initial and final navigation locations. Therefore, this GA-based controller doesn't need to be retrained when changes on these parameters occur.

The remains of this paper are organized as follows. Section 2 introduces the task domain and presents the simulated mobile robot Khepera. Section 3 discusses the implementation and describes the fitness function in details. Section 4 presents the simulation results and demonstrates the controller adaptability to some goals and environments. Finally, section 5 discusses some conclusions and considers possible future work.

---

* Khepera Simulator Package version 2.0: Freeware mobile robot simulator written at the University of Nice Sophia-Antipolis by Olivier Michel. Downloadable from the World Wide Web at http://wwwi3s.unice.fr/~om/khep-sim.html

2. The Task Domain

The GA controller system was developed to run with the Khepera Simulator. The Khepera Simulator is a freeware public domain software written by Olivier Michel [12]. This simulator allows writing control algorithms using C or C++ languages. A library of functions to drive the robot and display results is provided. The simulator runs on Unix workstations and features a nice X11 colorful graphical interface. Various different environments (called worlds) are available, although it is possible to design a new one from scratch.

The simulated mobile robot Khepera (see Figure 1) includes 8 sensors (small rectangles S0 through S7) allowing to detect the proximity of objects at its four sides : front, back, left or right. Each sensor returns an integer value between 0 and 1023. The value 0 means that no object is perceived, while value 1023 means that an object is very close to the sensor, almost touching it. Intermediate numbers give an approximate idea of the distance between the sensor and the object. The Khepera robot is also composed of 2 motors (large squares M1 and M2), each one being able to achieve a speed integer value ranging from –10 to +10.

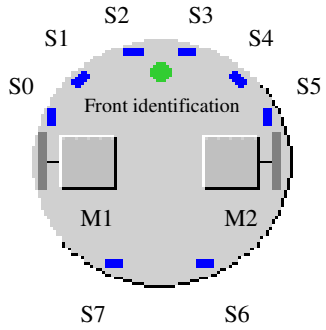


Figure 1. Simulated Khepera robot.

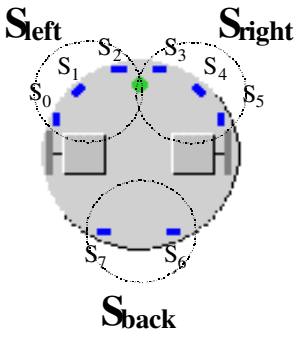
Besides the sensors and motors speed, the simulator also allows the reading of the position (x,y) and the direction's angle (α) of the robot in any location of the world. The x and y coordinates range from 0 to 1000 and the direction's angle α ranges between $[-\pi, \pi]$ degrees. Figure 1 shows the simulated robot looking to the front at $\alpha=\pi/2$ degrees. On the other hand, when the robot is looking to the right the direction's angle is at 0 degrees.

The main objective of this work was to implement a genetic algorithm routine to control this simulated robot. This controller drives the Khepera to reach the goal point, performing straight and collision-free motions, based on the input information provided by the sensors and the distance to the arrival point.

3. Implementation Details

The evolutionary procedure employed in the simulations consists in programming a standard genetic algorithm (GA) system, as described by Goldberg [9], to evolve the Khepera steps controller. This implementation is a new approach to the path-planning problem, representing each chromosome as a group of basic attitudes. These attitudes define the robot's movements in agreement with the feedback generated by its environment. Each feedback, which is used as input to the system, is based on the sensors reading and on the robot's direction to its goal location.

The sensors reading is presented to the GA system in a simplified form. Although the simulator provides 8 sensors values, it was verified that this number could be reduced without affecting Khepera's performance [2]. Figure 2 shows the simplification. It was observed that sensors S0, S1 and S2 assumed similar values when there was an obstacle in the robot's left side. The same happened with sensors S3, S4 and S5, and the pair S6 and S7, when the object was near the robot's right and back sides, respectively. Therefore, three "new" sensors reading were built, replacing the old ones as the inputs to the system and being able to detect obstacles at Khepera's left (S_{left}), right (S_{right}), and back (S_{back}) sides. These new sensors are calculated by the average of the corresponding original sensors, as Figure 2 illustrates.



$S_{left} = (S_0 + S_1 + S_2) / 3$

$S_{right} = (S_3 + S_4 + S_5) / 3$

$S_{back} = (S_6 + S_7) / 2$

Figure 2. Sensors reading simplification.

The other input to the system, the robot's direction to its goal location, is calculated as follows. Consider α as the direction's angle of the robot, provided by the simulator at any Khepera's location. Then, define β as the angle of minimum distance between robot's location and its fixed arrival position. Thus, the difference between β and α indicates if the goal point is in front of the robot, behind it, to its left or right sides, determining the Khepera's target direction. Figure 3 illustrates these considerations.

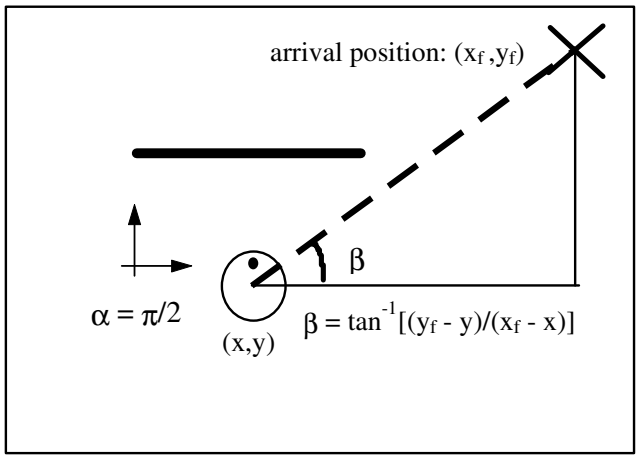


Figure 3a. The angle β points to the minimum goal's distance direction.

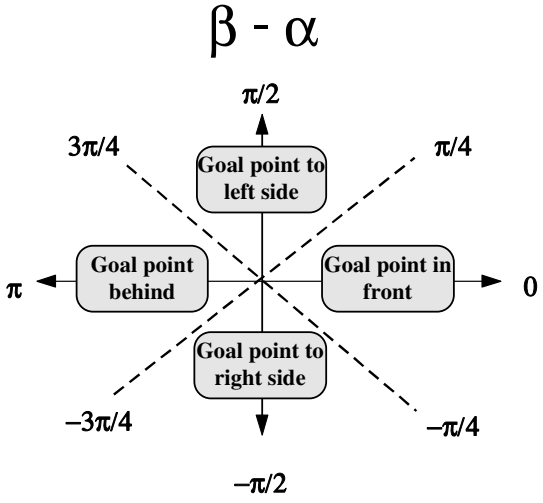


Figura 3b. Target direction of Khepera's movement.

Figure 3a shows the robot looking to the front at $\alpha=\pi/2$ degrees. The angle β is calculated by the trigonometric function $\tan^{-1}$ between Khepera (x,y) and arrival (x_f, y_f) positions. In this example, β can be supposed equal to $\pi/4$ degrees. Thus, the difference $\beta - \alpha$ is equal to $-\pi/4$ and this value indicates that the goal point is located at Khepera's right side (see Figure 3b). If, for instance, Khepera has been looking to the right at $\alpha=0$ degrees, the difference $\beta - \alpha$ would be equal to $\pi/4$ and its target direction would point to the robot's left side.

From the sensors reading and the robot's direction inputs, the Khepera's attitude can be defined according to the following states rule:

IF ((S_{left} > L) or (S_{right} > L) or (S_{back} > L))
THEN
    Obstacle detected
    Proximity-sensor = highest value (S_{left}, S_{right}, S_{back})
ELSE
    Obstacle not detected (collision-free)
    Target direction = $\beta - \alpha$
END

The constant L represents a collision threshold and may be determined between the sensor's range [0, 1023].

Each possible attitude defined by the states rule corresponds to a gene of the chromosome, and each gene is composed of the pair of Khepera's motor speeds (M1,M2). Therefore, the representation of the chromosome has the form presented in Figure 4.

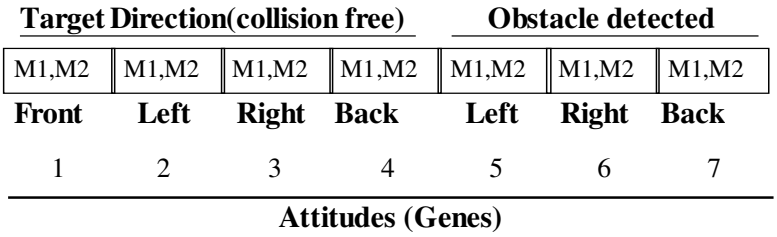


Figure 4. Chromosome representation.

As can be observed from Figure 4, the representation divides the chromosome in two parts. The first one determines the basic robot's target direction movements when no obstacles are detected (collision free states). The second part (obstacle detected states) considers the motor speed actions that avoid collision, being independent from the goal's location. Thus, if the robot's state is defined as collision free and its goal point is in its left side ($\pi/4 \leq \beta-\alpha \leq 3\pi/4$), then the Khepera's attitude to be carried out corresponds to the second gene of the chromosome. On the other hand, if Khepera detects the proximity of an object at its right hand side, it is going to take the action indicated at gene 6, without considering the target direction.

The fitness function F is calculated at each robot's time step and is composed of three components: V, D and A (see Figure 5). The first component (V) is maximized by speed, the second (D) by straight motion and the third (A) by action. The latter depends on the attitude defined according to the states rule. If Khepera's state is collision free and the arrival point is at its front (attitude 1), then the genetic algorithm (GA) should evolve the pair M1,M2 that minimizes this distance. On the other hand, if the robot is free but not oriented to the goal point (attitudes 2, 3 and 4), then it should find the pair (M1,M2) that rotates the Khepera's front in the direction of the target. It should be observed that in these attitudes (2,3 and 4) the straight direction component (D) is equaled to 1 to avoid contradictory learning (see Figure 5). Finally, if an object is located in the range of Khepera's view (attitudes 5,6 and 7), then GA should forget the goal location during some steps and find the best way (pair M1,M2) to avoid possible collision. The action component (A) is calculated by the average of the total steps executed per attitude because of the random noise (around 10%) added by the simulator in the amplitudes of the motor speed and sensors reading [12].

$$F = \sum_{i=1}^{7} (Vi * Di * Ai) \qquad ; 0 \leq Vi \leq 1; 0 \leq Di \leq 1 ; 0 \leq Ai \leq 1$$

$$Vi = (|M1| + |M2|)/(2 * Mmax) \quad , where\ |M|\ means\ absolute\ value\ of\ speed\ ranging\ [-10,10].$$

$$Di = \begin{cases} 1 & ,i = [2,3,4] \\ (1 + ((M1 + M2)/(2 * Mmax)))/2 & ,i = [1,5,6,7] \end{cases}$$

$$Ai = \begin{cases} 0 & ,TPi = 0 \\ (\sum_{t=1}^{TPi} AAi_t) \Big/ TPi & ,TPi \neq 0 \end{cases}, where\ t\ is\ the\ step\ index$$

$$AAi_t = \begin{cases} \Delta di/dmax & ,if\ (i = 1)\ and\ (\Delta di \leq 0) \quad ,else\ AAi_t = 0 \\ \Delta gi/gmax & ,if\ (i = [2,3,4])\ and\ (\Delta gi \leq 0) \quad ,else\ AAi_t = 0 \\ 1 - S_t/Smax & ,if\ i = [5,6,7] \end{cases}$$

Figure 5. The fitness function.

where

- i : attitude's index;
- Vi : attitude's speed ;
- Di : attitude's straight motion ;

- Ai : attitude's action;
- TPi : total of steps executed by attitude i ;
- AAi_t: action's fitness at step t;
- Δdi : difference of goal's distance between two consecutive steps;
- Δgi : difference of target's direction between two consecutive steps;
- S_t : proximity-sensor (S_{left}, S_{right}, S_{back}) highest activity at step t;

- dmax : maximum possible distance traveled by robot in only one step (limited by the simulator);
- gmax : maximum possible robot's angle rotation in only one step (limited by the simulator);
- Mmax : maximum robot's speed (equal to 10);
- Smax : maximum sensor's reading (equal to 1023).

The GA program was performed as follows. An initial population of individuals was created by assigning to each gene in the chromosomes a random new speed value ranging [-10,10]. The Khepera was left free to move as a result of the activity generated by the gene's attitudes, while its performance was recorded and accumulated according to the pre-designed fitness function (see Figure 5). Each individual could move for a limited number of steps and all their starting conditions were made the same.

When all the chromosomes in the population had been evaluated, three genetic operators - *selective reproduction, one-point crossover, and mutation* - were applied to create a new population of the same size. Selective reproduction consists of a linear scaling of the fitness values followed by a probabilistic allocation of a number to the offspring proportional to the fitness value of each chromosome. All offsprings, simple copies of their parents, were then randomly paired and a random one-point crossover was performed with a given probability. Each value of the newly obtained strings was then mutated with a given probability by adding a small random value within a negative and positive mutation speed range ([-10,10]). Thus, in this evolutionary system, the new population replaces the old one maintaining only the best individual (elitism). The process terminates after some number of generations, and the best chromosome represents the near-optimum attitudes found.

4. Results

In order to evaluate the system's performance, many tests were carried out, varying the GA's parameters. The best performance was achieved with the following parameters : population size equal to 100 individuals; crossover rate set to 80%; mutation rate set to 4%; constant L of collision threshold equal to 900; and total number of steps (n) equal to 300. The total number of steps (n parameter) was defined as bigger as possible, in order to test all robot's states without compromising the processing time.

The system takes no more than few seconds to evaluate each generation of 100 individuals. Furthermore, around the 50th generation the best individual usually dominates the population and further evolution can not be noticed. Thus, 25 runs of 50 generations were executed, saving the best chromosome of each run. Then, the best chromosome over all these runs was used to guide the robot in different environments, starting at different initial locations and with distinct goal points. When the robot reached the goal point, the total number of steps (T) were computed. Figure 6 shows four sample tasks in four different environments where, in each task, the near-optimal path obtained by the robot is displayed. It must be pointed out that the same best chromosome was tested in the 4 environments, without new retraining or evaluation. The different complexities of the environments and goal's location are reflected by the distinct T value (total number of steps).

As can be verified by Figure 6, Khepera tends to move towards the goal's location, traveling according to the minimum distance direction between the starting and arrival positions. When the robot detects an obstacle, it forgets the target's direction during some steps and tries to solve the problem of possible collision. Soon after this is solved and Khepera becomes collision free, it again looks for the goal's point, moving towards the minimum distance to the arrival location. This behavior is clearly observed in the second example (T=4000) of Figure 6. In this experiment, as soon as Khepera perceives an object in its front or left sides, it avoids the collision moving away from the objects, although these actions increase its distance from the arrival location.

These results are highly reliable and have been replicated in many runs of the experiment. Frequently, T = 1200 are sufficient to achieve good performance in most of the cases.

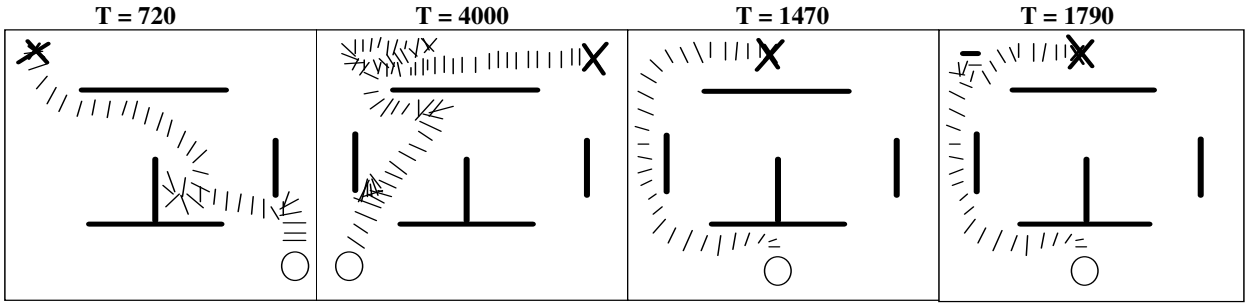


Figure 6. The controller adaptability to some goals and environments.

5. Conclusion

The experiments presented in this paper confirm the power of Genetic Algorithms (GAs) for solving multi-criterion optimization problems. Most importantly, unlike various learning techniques, GA can gradually evolve the control system of an autonomous robot by exploiting the variations in the interactions between the environment and the robot itself. This makes it possible to apply them to problems where neither the shape nor the size of the solution are known in advance.

The results obtained showed that the use of GA for the Khepera's controller produced extremely smooth path planning. The robot usually travels at full motor speed, successfully finds the diverse goal points from distinct starting points and environments and, at the same time, avoids obstacle and walls. It should also be observed that the robot's learning process does not depend on the environment, so that the robot doesn't need to be retrained if changes in starting point, goal point, and obstacle distribution occur.

The fitness function chosen had an important role in the success of this system. The fitness function was engineered in order to maximize multiple targets: the fast and straight navigation; the collision avoidance; and the location of the goal point. This was possible because these sub-goals do not represent contradicted meanings, allowing equal normalization. However, the strategy of evaluating each attitude (gene) of the chromosome in an independent way became limited in some situations. This limitation was characterized in some cases where the robot reached its target location after a clearly exaggerated total number of steps, getting almost stuck (second example of Figure 6). This can be justified by the fact that Khepera has been modeled to forget its goal location if it encounters a collision dangerous area (area with obstacles) and, consequently, being able to move away from its arrival point in such cases. Also, the fitness functions corresponding to collision detection attitudes have not been well refined, analyzing only the highest activity sensor whatever side it belongs to.

Therefore, exploration of evolutionary diverse strategies, genetic representations, genetic operators, and fitness functions for designing robust and large freedom degrees behavior controllers for robots under a variety of goal's performance and environment constraints is the subject of ongoing and future research.

Acknowledgment

This work was partially supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil.

References

1. Balakrishnan, K. and Honavar, V.; *Analysis of neurocontrollers designed by simulated evolution.*. Proceedings of the International Conference on Neural Networks, Washington, D.C., 1996.

2. Balakrishnan, K. and Honavar, V.; *On sensor evolution in robotics.* Proceedings of the First International Conference on Genetic Programming, Standford University, CA, pp. 455-460, 1996.

3. Balakrishnan, K. and Honavar, V.; *Some Experiments in Evolutionary Synthesis of Robotic Neurocontrollers*. Proceedings of the World Congress on Neural Networks (WCNN'96), San Diego, CA, September 15-20, pp. 1035-1040, 1996.

4. Baluja, S.; *Evolution on artificial neural networks based autonomous land vehicle controller*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 26, no. 3, pp. 450-463, June 1996.

5. Brooks, R. A.; *Artificial life and real robots.* Towards a Practice of Autonomous Systems : Proceedings of the First European Conference on Artificial Life edited by F. J. Varela, P. Bourgine. Cambridge, MA : MIT Press/Bradford Books, 1992.

6. Davis, L.; *Handbook of Genetic Algorithms*, VNR Comp. Library, 1990.

7. Dorigo, M. and Schnepf, U.; *Genetic-based machine learning and behavior based robotics : A new synthesis*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 23, no. 1, pp. 141-154, 1993.

8. Floreano, D. and Mondada, F.; *Evolution of Homing Navigation in a Real Mobile Robot*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 26, no.3, pp. 396-407, June 1996.

9. Goldberg, D.; *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

10. Hoffmann, F.; *Evolutionary Learning of Mobile Robot Behaviors,* the First Workshop on Frontiers in Evolutionary Algorithms FEA'97. Downloadable from the World Wide Web at http://HTTP.cs.Berkeley.EDU/~fhoffman

11. Koza, J. R.; *Genetic Programming. A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems.* Technical Report STAN-CS-90-1314. Stanford University Computer Science Department, 1990.

12. Michel, O.; *Khepera Simulator Package,* version 2.0: Freeware mobile robot simulator written at the University of Nice Sophia-Antipolis by Olivier Michel. Downloadable from the World Wide Web at http://wwwi3s.unice.fr/~om/khep-sim.html.

13. Nolfi, S. et al.; *How to evolve autonomous robots : different approaches in evolutionary robotics*, Proceedings of the Fourth WorkShop on Artificial Life, 1994.

14. Xiao, J. and Zhang, L.; *Adaptive Evolutionary Planner/Navigator for Mobile Robots*, IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 18-28, April 1997.