

Pinky: A Reflex-Based Agent For The CiberRato Competition

Luis Rei

Faculty of Engineering, University of Porto, Portugal
Luis.rei@gmail.com,
WWW home page: <http://luisrei.com>

Abstract. Pinky is a reflex-based agent whose purpose is to navigate through a labyrinth in order to find the beacon in the simulated robot competition CiberRato. This article presents the strengths and problems of using a purely reflex-based agent in that competition. Finally some solutions to the problems are presented.

1 Introduction

CiberRato [1] is a simulated version of the MicroRato robotics competition, held by the University of Aveiro. MicroRato is a competition between small autonomous robots whose primary purpose is to navigate a labyrinth from a starting point to the beacon. The secondary purpose is to return to the starting point. In CiberRato each simulated robot has a standard virtual body including sensors and actuators.

Pinky is a software agent that controls the virtual robot and attempts to have it reach the virtual beacons position. Pinky is a simple reflex agent and thus acts only on the basis of the current percept. Hence the environment Pinky operates in can be classified as partially observable, deterministic, sequential, static, continuous and single-agent.

This article summarizes the main characteristics of Pinky, the problems encountered in solving real CiberRato labyrinths, the solutions implemented to solve those problems and proposes other solutions.

2 The Virtual Robot

The virtual body of the robot includes several sensors: Infra-Red obstacle sensors, collision sensors (bumpers) a beacon sensor which gives the direction of the beacon and a ground sensor which detects whether the robot has reached the beacon.

Several actuators are also available: two motors each controlling a wheel and 3 signaling lights (LEDs).

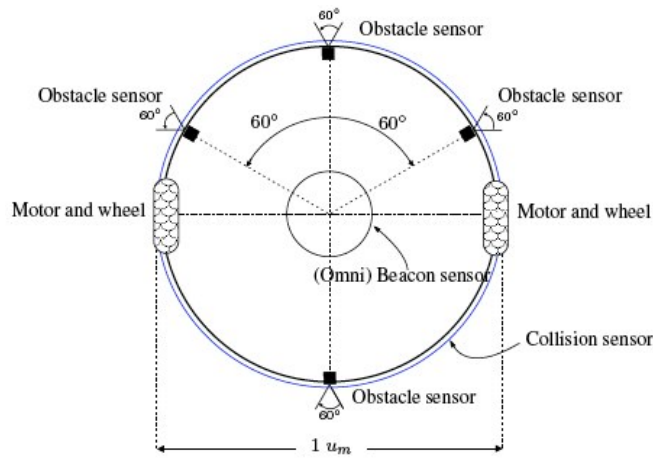


Fig. 1. Virtual Robot Schematic

3 The Pinky Software Agent

The Pinky software agent that controls the virtual robot makes use of the three frontal obstacle sensors, the beacon sensor and the ground sensor. In pseudo-code, the basic agent function is:

```

IF reached beacon
  THEN END;
ELSE IF wall in front
  THEN reverse direction
ELSE IF wall to the left
  THEN turn right
ELSE IF wall to the right
  THEN turn left
ELSE IF beacon IS visible
  THEN turn beacon direction
ELSE
  keep going in the same direction

```

An agent function such as this, based on condition-action rules using only the current percept, describes a simple reflex agent. This class of agent is easy to implement and has very modest memory requirements and processor usage. Thus they can be developed very quickly and would be cheaper to build due to the hardware requirements if they were to be used in non-simulated robots.

4 Problems

The biggest downside of simple reflex agents in this case is that in labyrinths that require the robot to move away from the goal without being able to move along a wall, they will be unable to reach it. These can be divided into two types: those which require the robot to do this for a very small distances such as the one in Fig. 2 and those that require the robot to do it for long distances such as the one in Fig. 3.

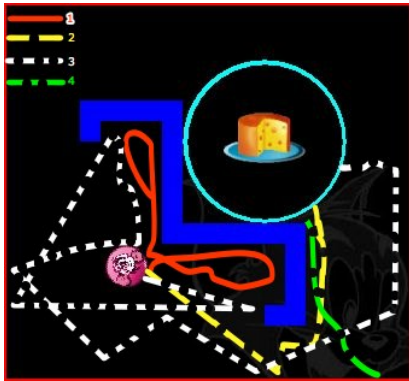


Fig. 2. A part of the labyrinth used in the CiberRato 2001 Final. The small edge of the walls make it impossible for a purely reactive agent that is too close to the wall to reach the beacon.

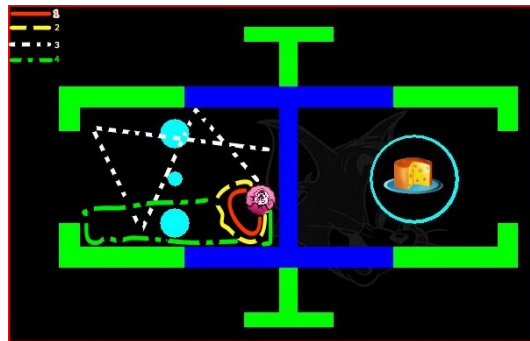


Fig. 3. A large portion of the labyrinth used in the CiberRato 2003 Final. It requires the agent to move away from the beacon for a long distance.

5 Proposed Solutions

The second version of Pinky performs random actions with some small probability whenever there is no danger of collision. This approach was successful in solving the first type of problematic labyrinths.

In order to solve the second problematic labyrinth a more complex approach is required. A third version of Pinky uses an algorithm which resembles the reverse of simulated annealing. As time goes by, entropy increases. The higher the entropy, the more likely it becomes that Pinky will chose a random action. Unfortunately, this approach will, in many labyrinths, decrease the speed at which Pinky reaches the beacon and even worse, because the time to reach the beacon is limited, Pinky may still not reach it at all in the labyrinths for which this approach was specifically developed to deal with.

A fourth version of Pinky follows labyrinth walls. Wall-following behavior can be implemented differently: the agent can follow walls to their edges and then try to follow the beacon or it can always follow the walls. Pinky uses the first approach. The second approach fails whenever the labyrinth wall the agent is following isn't directly connected to the beacon area or when the wall is too far away from the beacon, such as in the labyrinth shown in Fig 3.

Other possible solutions include using a previously trained Neural Network, adding memory to Pinky or turning it into a Model-based agent or any other class of intelligent agent.

6 Results and Discussion

In non-problematic labyrinths, Pinky often moves quickly towards the beacon but not always because turning towards the beacon can be the wrong chance in some situations while in others the order of the rules in the agent function, may make it turn right where left is an equally valid choice as well as better.

The second version, occasionally performing random actions, did not significantly increase the time it took to reach the beacon in simple labyrinths but it also did not guarantee that the beacon would be reached.

The third version introduced to solve the more difficult second type of problems, making the probability of random actions increase with time, can have a very negative impact on the amount of time it takes to reach the beacon in simple, non-problematic labyrinths making it possible that even in such labyrinths the beacon will not be found within the time limit. Experiments showed that the performance is very poor and obviously unpredictable due to the use of random actions. Usually the agent is unable to find the beacon in the second type of problematic labyrinths and when it does, it takes a very long time. Experiments also showed that it is possible to improve the results by optimizing the values for the maximum entropy and for the rate at which it increases. However, while it may be possible to chose good standard values, optimal values would be specific to each labyrinth.

The fourth version provides more predictable results but unfortunately also fails to find the beacon in certain types of labyrinths.

References

1. University of Aveiro: CiberRato 2008 Rules and Technical Specifications. (2007)
2. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. (1995) 1013–1015