Comparison of two methods for solving Markov Decision Processes in the persecution-evasion problem

Michel García†, Cinhtia González†, Enrique Succar†† and Eduardo Morales††

† Universidad Autónoma de Yucatán-Unidad Tizimín, †† INAOE

Summary
There are two basic approaches to solve Markov decision processes (MDP). One is to build a model of the process and to obtain the optimal policy using value or policy iteration. The other consists in obtaining the policy by trial and error using reinforcement learning. Although the two have been used to solve different decision problems, their merits and limitations have not been compared experimentally in the same domain. We have used both approaches to solve a pursuit-evasion problem in mobile robotics. We represent this problem as relational MDP, considering the distance and position of the evader in relation to the pursuer; and obtain the optimal policy for the pursuer by: (i) building a model and solving it with value iteration, and (ii) by using reinforcement learning. We have implemented both approaches in a simulated environment and compared them in terms of effectiveness, efficiency and ease of model construction.
Key words:
MDP, Reinforcement Learning (RL), Value Iteration.

1. Introduction

The persecution-evasion activity is one of the fundamental problems of robotics and it consists in accomplishing the tracking of a robot to another one. The before mentioned problem can be represented like a Markov Decision Process (MDP) and resolved of several forms. A comparative analysis among two they're made in this document: By means of the iteration value and with a reinforcement learning approach. Later time both methods are described.

Basically, the problem consists in accomplishing the persecution until the evader robot is reached. The above can turn out well of various manners, which here are implemented and compared only two. The first consists in providing the robot a model of his environment, as from the one that the best politician may obtain movements as from, depending on the status in the fact that he meet. Second form consists in accomplishing autonomous movements that the same pursuing robot must go learning on one's own, unless somebody say it specifically what to do or how to do it, but only based on the feedback that he is given by means of certain dynamic rewards, associates to realized actions.

Both forms were implemented under similar conditions, in a simulation environment on that various tests were performed and are described later.

2. Markov Decision Problems

Markov Decision Process [12] is a model that it lets to act in a process of decision in the time, acquaintance's representation also like process of sequential decision. By means of this model it is possible to represent problems which one must drink in decisions be more than enough what action realizing, with the aim of minimizing the cost correlated to a series of interactions with the environment.

Several methods exist to resolve a MDP, that is, to find the best action associated to each status, that in aggregate they will lead to achieving, of optimal manner, the desired goal. Such methods can be classified in three categories: Dynamic programming methods, differences temporary methods and reinforcement learning. The iteration value corresponds to the first of these, and along with reinforcement learning, are an important part of the development of the present work, so both are described hereinafter.

2.1 Iteration value

The classical method to resolve a problem formulated as a MDP is known like value iteration. The basic idea consists in calculating the profits out of every possible status and at a later time to use them to select the following action to realize. As his name suggests it, he is an iterative method where the status utility depends in of the sequence of actions taken to start of this state, according to the established policy. This method is used when the model of the environment is known, i.e. the array state transitions probabilities, and expected values of rewards.

Initially, the states utility can be obtained like the utility expected of all the possible sequences of stock. Once separability's condition was given, the utility of a status can be obtained in iterative form maximizing the utility of the next state.

Manuscript received April 5, 2010
Manuscript revised April 20, 2010

2.2 Reinforcement learning

Reinforcement learning is a method that permits the autonomous robots learning from its own surroundings and besides, learning while they find themselves immersed at the aforementioned surroundings. This method has had an important heyday of late years and reaches out for to achieve that an autonomous agent that receives sensorial information and acts at a surroundings may learn how to elect optimal stock to achieve his objectives. This generic problem covers up tasks such like learning to control a movable robot, learning to optimize operations at factories, or learning how to play board games.

In each moment the agent executes an action, to the one that he is provided a reward or a punishment to indicate the desirable that the new status that the aforementioned action led to proves to be. The agent's task is to learn from rewards or punishments that he receives when electing sequences of actions that they generate a bigger positive reinforcement, and of indirect manner, attaining of efficient manner his goals. [9].

He has to do with, therefore, a learning supervised system that receives information of the external and a "teacher" that produces positive rewards or refusals according to the quality of realized actions. The reinforcement or reward is once show of the action accomplished, once a previous status was given was hit on. That is, if it takes place to a status better than the previous, then a positive reward is granted, otherwise penalizes him the action that he carried into that state.

3. Implementation

In order to implement both methods described previously, they defined certain statuses and stock, which describe themselves from now on. At a later time the procedure used for realized implementation is detailed.

3.1 States

In the environment simulated navigational, the robots move for all the space, it as the place opens into a set of states that the spectator can observe and whose appearance is similar to the one that shows in the Fig. 1.

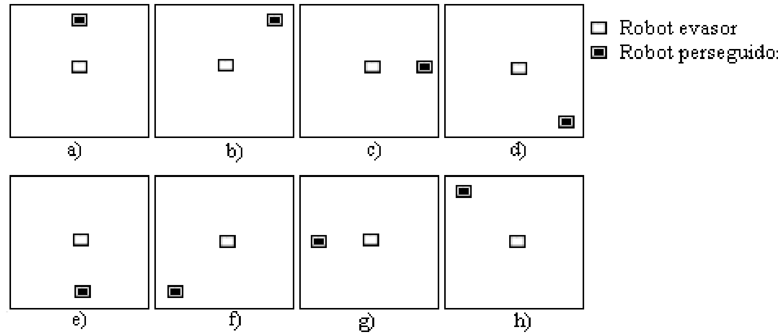


Fig. 1. Examples of real states in the environment. Different positions relatives, evader robot in relation to the persecutor.

However, the sensors are the only one midway for the robots to get information from his environment, that provide information on the objects that they find themselves visible in his reach themselves. Due to the above, and with the purpose of working with a number of states easy to handle, the states to consider are determined from certain components, applying the concept of abstraction. The said components look in the Table 1.

Table 1 Description of states components.

Component	Description	Values
Distance	The evader robot in respect of the pursuing robot separates the one to which the robot is found. It is determined according to the size of the observed image.	Big Medium Small Empty
Direction	Direction that evader robot finds himself. It is determined according to the position of the image perceived at the camera.	Left Right Center

Distance. Indicates the distance the robot is found, and it is determined taking into account the size of the image observed at the camera. The Big parameter means that the image observed at the pursuing robot's chamber is big, which as it means that the distance that this meets to is short.

Direction. The position of the evader robot in respect of the pursuing robot. The values Left, Right and Center indicate the position in which the objective is found, and therefore, the movement that the robot must accomplish to maintain his contender's image in the center of the camera.

In a environment without obstacles, the observations of the robot are determined only for distance and direction to which the objective is visualized. The Fig. 2, illustrate these observations, that the states act for considering resolve the presented problem.

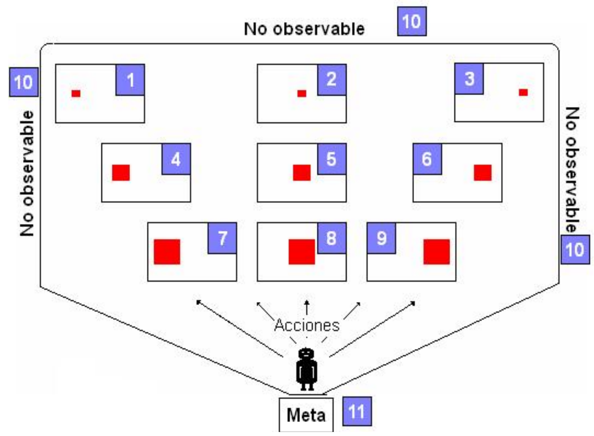


Fig 2. Basic possible states for the robot. Are 11 states the ones that 9 (of the 1 to the 9) happen of of distance-direction combinations, an only state (10) for all those positions in which the objective is not found visible at the camera and that an additional state considers itself the goal in considers itself additional (11).

Basic states are those were shown in the Figure 2, and they suit their purposes like base to generate the states endings considered in the problem, that they consist of a couple of basic states: The previous and the current. Therefore, the number of considered states is of (11)2 = 121 states.

3.2 Actions

Actions are those movements that can be accomplished in each status to go by to another one. They were considered five actions: Turn left, Partial turn left, advancing, partial turn right and Turn right.

3.3 Value iteration Implementation

On the value iteration implementation, the probability array of transition got directly from the simulated environment. In order to accomplish this, allows robot to explore the environment of manner pseudo-aleatory during certain time (specified at the section of experiments and results).

In turn, they generate their arrays of rewards, which create themselves following the judgment:

- In order to meet the goal: +1
- Toward an obstacle (empty status): -1
- In order to step forward: +0.1

At a later time, the MDP is the algorithm of valuable repetition, whose parameters resolvedly intervening are: The probability array of transition just described odds, the matrixes of rewards and a factor of decrement of 0,9, which determined itself on the basis of experimentation realized previously . Later on the behavior of the robot using the aforementioned policy can be observed, at the section of comparative analysis.

3.3 Reinforcement learning Implementation

The learning algorithm for reinforcement proposed by Watkins 11 was implemented. Basically, this algorithm consists in finding all of the possible statuses for certain stock and to take a their record. For each executed action he grants a reward that can be positive or negative, with the end of than in following visits to the same statuses may determine him which one is the best-suited action.

The execution was accomplished during different spans, specified at the section of experiments. During his performance, the pursuing robot does a journey of the environment attempting to find the evader robot and all states are registered for the ones that he transits for, just like the actions that he sells off.

The strategy of the evader robot restriction consists to sail at random across the environment, with the one and only of that, with help of his chamber, when visualizing and identifying his adversary accomplish a spin in the deeply felt opponent and where he have bigger space to navigate, with the aim of losing sight of it and procuring thus not to be reached. In realized implementation he takes a record of actions taken, which step by step they conduct which to an optimal behavior of the robot.

4. Experiments and Results

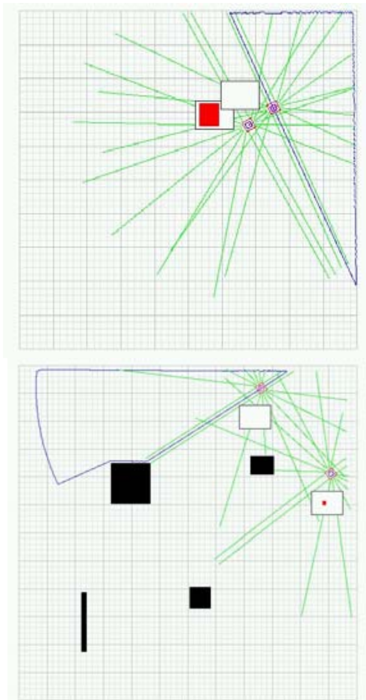
Preliminary tests were accomplished using a blobfinder, Player's device whose show is to detect colors with a camera. From now on experiments accomplished, in environments with obstacles and without them, with each one of implemented methods are described.

In the implementation of the reinforcement learning algorithm, during the first minutes of execution the tax evader once has identified it notices that to go after the robot turns out to be to the pursuing robot difficult to him. However, as the time passes, the robot is more and more able to accomplish his adversary's tracking and also navigates of more docile and soft manner.

In the implementation of the algorithm of iteration value, all along it is noticed that the robot chases its objective, with movements very well defined even to reach it finally to the evader robot.

Stroke of ball mentioning than, in addition to the policy of given actions the solution the fact that which generates the algorithm, the same strategy of evasion of obstacles in the pursuing robot was incorporated, also is present in learning for reinforcement, so that they consider equitable situations equitable situations.

The Fig. 3 shows the performance of the algorithm in Player/Stage in an environment with and without obstacles.
.



a)

b)

Fig. 3. The following's image sold off in Player/Stage in an environment with (b) and without (a) obstacles.

5. Comparative analysis

In order to accomplish a comparative analysis of quantitative nature was proceeded from the following manner: First it was allowed to navigate the robot during incremental time intervals with each one of implemented algorithms. At a later time the policy generated with each method was compared. As it was mentioned previously, the performance of the algorithm of valuable repetition practically does not vary once the optimal policy has been generated; However, for effects of equality of situations, the time was considered that him important it is allowed to navigate the start to the robot construct his array of transitions probabilities, that will be useful for like piece of information the obtaining of the policy.

The results obtained when comparing both policies are in the Table 2. The Table 3 shows the average time that the robot spends in achieving your purpose after 60 and 120 minutes delays, learning and exploration, respectively. Was realized 10 executions with each method.
With the iteration value, the average time the fact that the pursuing robot in catching up the evader robot with a previous inferior exploration to 1 minute delays becomes of approximately 30 seconds, and with an exploration of 30 minutes delays 24 seconds. That is, the difference is minimal.

Table 2. Similarity in policies obtained after 30, 60, 90 and 120 minutes of exploration of the environment. The number of states with the same action with different action, or states visited, in RL in relation to value iteration are shown (VI). The percentages respects to 121 states are given.

State-action	30'	60'	90'	120’
Same	9.9%	14.1%	15.7%	16.5%
Distinct	10.7%	13.2%	14.9%	16.5%
Not visited	79.4%	72.7%	69.4%	67%

In the previous table it observes an important quantity of statuses visited by the algorithm RL, it as he proves to be congruent because some states are necessary to they are practically impossible to visit if it is considered that it is not like base accomplishing certain transitions, such from state 1 to the goal.

Table 3. Average time that the robot spends in achieving his purpose after 30, 60, 90 and 120 learning minutes (RL) and exploration delays (VI), respectively. Was executed 10 executions with each method and are reported the times in seconds.

Approach	30'	60'	90'	120’
RL	58.6”	50.5”	47.3”	35.2”
VI	24.2”	23.9”	23.3”	23.2”

6. Conclusions and Future Work

After achieving the objectives initially presented with both methods, it is possible concluding than reinforcement learning, prove useful because the agent does not call for specific instructions, rather one is dedicated to learn on the basis of test and error. To the start the performance, prove little efficient, but step by step he goes improving in agreement lapses the time and his archives of information go away filling with information.

In the case of the modeled by means of MDP, it could become verified that to for to achieve an optimal performance, a good training phase is necessary, as well as a correct definition of states and actions to realize . With this method it is possible to accomplish a more efficient following, which observes in Tables 2 and 3 showed previously.

As future work, both algorithms will be implemented in real robots.

7. References

[1]. Isler, V. Kannan, S. Khanna S. "Locating and Capturing an Evader in a Polygonal Environment"

[2]. Garía, R. Batlle, J. Magí, Ll. Pacheco, Ll. "Seguimiento de multiples objetos: un enfoque predictivo" Universidad de Girona, Dep. de Electrónica, Informática y Automática.

[3]. Pomares, J. Gutiérrez, R. López, J. M. Torres, F. Gil, P. "Seguimiento de trayectorias 3d mediante control Visual basado en imagen", Dpto. Física, Ingeniería de Sistemas y Teoría de la Señal. Universidad de Alicante.

[4]. Ruckelidge, W. J. "Efficient Computation of the minimum Hausdorff Distance for Visual Recognition", Phd thesis, Cornell University, 1995. CS-TR1454.

[5]. Sánchez, E. Hernández, M. "Seguimiento de Objetos Móviles usando la Distancia de Hausdorff", Universidad de Las Palmas de Gran Canaria.

[6]. Palomares, J. Torres, F. "Control visual basado en flujo de movimiento para el seguimiento de trayectorias con oclusiones", Universidad de Alicante, España.

[7]. Sachs, S. LaValle, S. Rajko, S. "Visibility-Based Pursuit-Evasion in an Unknown Planar Environment" The International Journal of Robotics Research Vol. 23, No. 1, January 2004, pp. 3-26.

[8]. Tovar, B. LaValle, S. Murrieta, R. "Optimal Navigation and Object Finding without Geometric Maps or Localization".

[9]. Mitchell, T. Machine Learning. McGraw-Hill, 1997.

[10]. Vaughan, R. Gerkey, B. A. Howard. "On device abstractions for portable, reusable robot code" Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 21212427, Las Vegas, Nevada, October, 2003. Player is freely available under the GNU General Public License from http://playerstage.sourceforge.net.

[11]. Watkins, C. Learning from Delayed Rewards,Thesis,University of Cambide, England. 1989.

[12]. Puterman, Martin. *Markov Decision Processes.* John Wiley & Sons. New York, 1994. 672 pp.

[13]. Morales, Eduardo. Capítulo 11: Aprendizaje por refuerzo. Notas del curso "Búsqueda". 2004. http://dns1.mor.itesm.mx/~emorales/Cursos/Busqueda04/refuerzo.pdf. Ultimo acceso realizado el 8 de mayo de 2005.

[14]. Richard S. Sutton, Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation & Machine Learning S.).* MIT Press. 1998. 338 pp.

[15]. Puterman, Martin. *Markov Decision Processes.* John Wiley & Sons. New York, 1994. 672 pp.

[16]. Russell, Stuart; Norvig Meter. *Inteligencia Artificial, un enfoque moderno.* Editorial Prentice Hall. 1996.

[17]. Markov Decision Process (MDP) Toolbox v1.0 for MATLAB (INRIA) http://www.inra.fr/bia/T/MDPtoolbox/

[18]. T. Mitchell. Machine Learning. McGraw-Hill, 1997.

[19]. Notas del curso Razonamiento con Incertidumbre, Dr. Enrique Sucar. http://dns1.mor.itesm.mx/~esucar/incertidumbre.html.

[20]. Proyecto Player/Stage. http://playerstage.sourceforge.net/

Cinhtia Maribel. Master in Computer Science for the Institute Technology of Monterrey in México, is professor of the Autonomous University of Yucatán, Actually responsible of the project: "Estrategias para obtener aprendizajes sifgnificativos utilizando tecnología robótica". His researcher lines: Optimization, Artificial Intelligence, Mobile Robots.

Michel García. MC in Computer Science for the Institute Technology of Monterrey in México, is professor of the Autonomous University of Yucatán, Actually collaborate in the Intelligent Systems Lab. His researcher lines: Machine Learning, Artificial Intelligence, Mobile Robots.

Enrique Succar, Ph.D. in Computer Science Philosophy for the Imperial College of Science, Technology, and Medicine, London, England, is professor of the National Institute of Astrophysics, Optics and Electronics.
Has been published more than 100 papers, his researcher lines: Artificial Intelligence, robotics, artificial vision

Eduardo Morales. Ph.D. in Computer Science for The Turing Institute – University of Strathclyde, Glasgow, U.K., Researcher in Computer Science and Professor of the National Institute of Astrophysics, Optics and Electronics. Actually responsible of the project: Learning by imitation with humanoids. His researcher lines: Machine Learning, Artificial Intelligence, mobile robots.