CMP9132M Programming an AI with Stochastic Actions to solve a MDP problem in a partially and fully observable environment

UNIVERSITY OF LINCOLN

line 1: 2nd Stephen Rerri-Bekibele
line 2: *School of Computer Science. University of Lincoln (of Affiliation)*
line 3: Lincoln, United Kingdom
line 4: 16663359@students.lincoln.ac.uk

Table of Contents

***Abstract*— Programming an AI to navigate an environment is a very common occurrence in the gaming world. There are many different path finding algorithms [2] and ways this has been done. From A*[3] to Dijkstra's algorithm [4]. The aim of this research paper is to take on such a task and use an MDP [1] approach to program an AI to survive as-long-as possible in arenas of varying sizes, whilst also collecting bonuses in a non-deterministic World simulation. This is repeated for a partially observable world as well as a fully observable world while the AI is also dealing with Stochastic actions [6]. The outcome of this research paper proves that an AI is more capable of effective planning and surviving longer when using MDP solutions in a fully observable scenario than it is in a partially observable scenario.**

Keywords—MDP, survive, deterministic, planning, solutions, observable, Stochastic actions (*key words****)***

Introduction

In order to talk about implementing a path finding algorithm we must first look at some useful terminology and words which are integrated into the decision-making process. More commonly referred to as: Complex decision making [5]. Complex decision making is all about linking decisions together. This is extremely crucial when one decision leads to another; and each decision depends on the ones before and affect the ones after [6]. Making informed decisions requires knowledge about the world around us. This is true for any time of world whether real or simulated. The level of this knowledge is discretely referred to as fully observable or partially-observable in the context of world information.

A "fully observable" world is one where we always have access to all the information about the world "all the time". A "partially observable" world is one where we do not have access to all the information about the world all the time (Simon, 2022). We may start off with some information, the rest of which may be acquired over time via exploration, or another means. However, at no point can we have all the information about the world all the time, unlike a fully observable world. This is significant because it means a partially observable world is always limited by its view of the world. Within the Complex Decision-making process, Deterministic and Non-Deterministic are terms used to refer to the probability of the consequences of one's actions. In a Deterministic world, there is only one consequence no matter what action is taken; therefore, the probability of a desired consequence for of any given action is 100%. In a non-Deterministic world, the probability of a desired consequence for any given action can be between 0-100%. This reasoning is referred to as Stochastic probability [7].

Concept

To solve this task, I chose to use a Markov Decision Process via a python library called the MDPtoolbox [12]. A Markov decision process (MDP) is A Discrete-time stochastic control process. It is used to provide a mathematic framework for modelling decision making in situations where outcomes are partly random and partly under the control of a decision maker [8]. This means the control is stochastic which is the case for our AI's actions as previously described. This was why I thought the MDP would be an effective solution to solving the maze task.

"The MDP relies on the notion of state (S), describing the current situation of the agent, action (A) affecting the dynamics of the process and reward (R), observed for each transition between states" [9]. With the knowledge of the stochastic decision process and the AI's state at every time step, the MDP's goal is the survival of the agent for as long as possible while collecting rewards. To solve this task, phrasing the task as an MDP problem thus means searching for a policy, in a set, which optimizes a performance criterion for the considered MDP. This policy is then used to direct the AI by giving it the best action for each state of the grid maze it is in [6].

State	0	1	2	3	4	5	6	7	8	9
0	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
1	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
2	-0.04	-0.04	-0.04	-0.04	-1	-0.04	-0.04	-0.04	-0.04	-0.04
3	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
4	-0.04	-1	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
5	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
6	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
7	-0.04	-0.04	1	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
8	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	1
9	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04

The example above is a replica of the reward matrix used in this research and illustrates that there are positive and negative rewards i.e., +1 and -1, for being in certain states of the grid such as the positive rewards in states (8, 9) and (7, 2). In non-terminal states, the reward is set as -0.04. By assuming that the utility of a run is the sum of the reward states, the -0.04 is an incentive for the AI to take fewer steps to get to the terminal state [11].

For the base grid of size 10 by 10 used in this project, the MDP requires as input, two matrices of data, one is a Probability matrix A.K.A transition model P (s'| s, a), of shape (A, S, S) where A are actions and S are states. This specifies an array of possible actions (A) for each state. Each S x S, then specifies the transition probabilities of reaching the second state by applying that action in the first state [6]. This meant the probability matrix would have been a (4,100,100) matrix because there were 4 possible actions the AI could take at any state.

The Probability and reward matrices can be generated recursively using the bellman equation [10] to compute the expected utilities for each state of the maze grid.

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^{\pi}(s').$$

Figure 1. The Bellman Equation used to compute the probability value of each cell for a change of state by a given action

A policy much like:

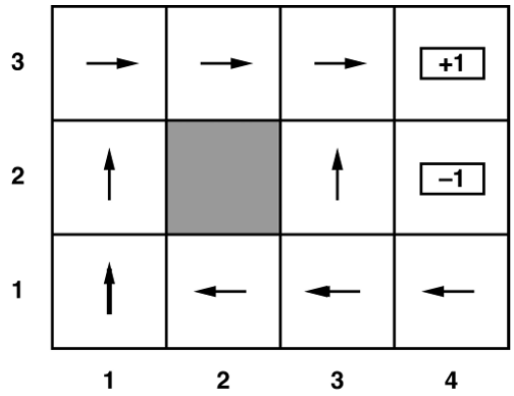

Figure 2. An example of a simplified policy that will be generated by the MDPtoolbox library from the Transition model and Rewards function.

is then generated from the two tables, and this then gets the best possible action in each state and returns that value as a command to the AI when it is in each of those states.

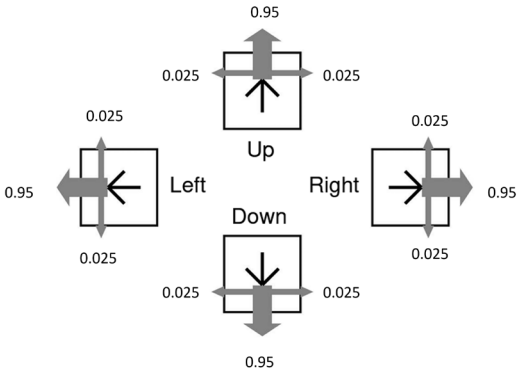

Figure 3. A motion model to illustrate the stochastic nature of the AI's actions

95% of the time the AI moved as intended and 0.5% of the time it moved perpendicular to the direction indented. Half the time to the left and half the time to the right.

$$P(s'|s,a)$$

Where a is the action that takes the AI from s to s'. These transitions are assumed to be (first order) Markovian and therefore only take into consideration, the current and next states.

In a special case such as whenever the AI hit a wall, a "wall bouncing algorithm" had to be developed. This algorithm simply returned the probability of the action in the current state plus the probability of taking the action as the value of the current state.

This method was chosen because it allowed the AI to dynamically switch between two states, one where it moved with a policy that told it the best action for any state the AI was in and another that allowed the AI to successfully navigate to bonuses whilst avoiding meanies and pits coming in and out of visibility limit.

EVALUATION

Surprisingly, the AI didn't always perform better when the partially observable parameter was set to FALSE. This is evident because the MDP was not perfect and could not always provide a better policy to follow before too many obstacles were present as new meanies were added after a few time steps of the simulated world.

Three scenarios were created with varying amounts of bonuses, pits, arena sizes and speed of spawning meanies. For the sake of word count only the 10 by 10 grid will be analyzed in detail and the result tables in the appendix will show the results of the other tested parameters.

When partially visible was TRUE, the first scenario had a mean score of 14.6 and survival time of 9.6(s) and the second scenario which has double the parameters of the first scenario had a mean score of 26.5 and survival time of 21.4(s). Lastly, the 3rd scenario which yet again doubled the parameters had a mean score of 15.8 and survival time of 14.3(s).

In contrast, when partially visible was FALSE, the first scenario had a mean score of 15.6 and survival time of 11.2(s). The second scenario had a mean score of 16.9 with survival score 7.6 and lastly, the third scenario had a mean survival time of 19.7 and survival time of 10.3.

At the time of this test, this demonstrated that in most cases, a larger grid size led to longer survival times and a higher bonus. In smaller grid sizes the AI was getting

caught more quickly, resulting in a lower score and survival time. Tables of results can be found below in the Appendix.

In conclusion, my method of generating a reward matrix via the bellman equation fell short because it recursively added each reward but only one at a time for the arrays of bonuses, pits, and meanies. I therefore had to create a priority queue whereby only the instances of each obstacle or reward closest to the AI at every time step would be added to the reward matrix and dealt with at each time step. This meant that when dealing with multiple meanies (Figure x) the AI could get caught with a pincer move because only the closest meanie chasing it would have a negative reward associated with it and thus it would not see another meanie coming from another direction until it was too late. In addition, this meant that the AI would not perceive obstacles as "close" if there was a meanie with coordinates closer to the AI than the adjacent obstacle

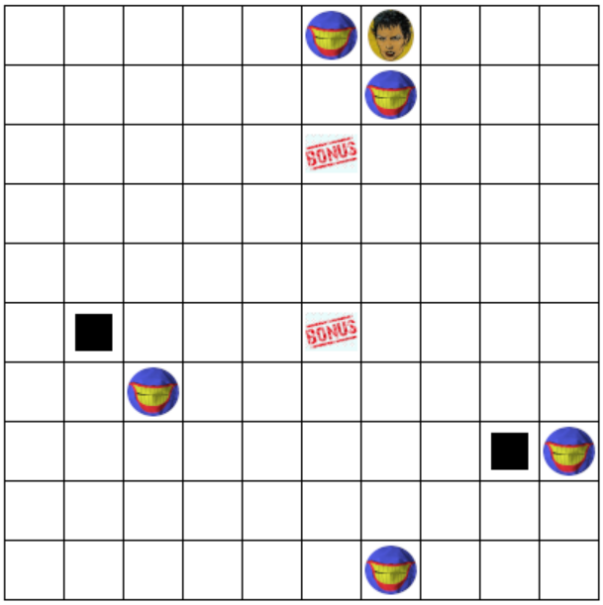

Figure 4. An image of the grid showing how Tallon can get trapped by multiple meanies. In this example the policy gave the best action for that state and got Tallon out to survive for longer.

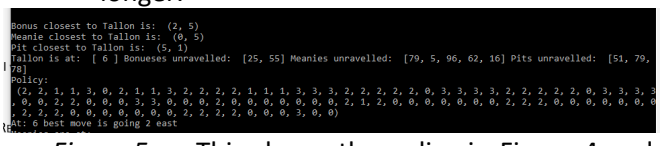

Figure 5. This shows the policy in Figure 4 and how Tallon interprets the closest obstacles.

This shows that although the meanies at (1,6) and (0,5) were equally close to Tallon if we look at the graph, Tallon still sees the closest Meanie as the one at (0,5) since Tallon is at (0,6) and this is dangerous because it means Tallon could have moved into the adjacent meanie at (1,6).

Suggestions for improvement include finding a way to add all current meanies, pits and bonuses into the reward function so the MDP library can create a better policy and making the code more programmer friendly.

ACKNOWLEDGMENT

I would like to acknowledge Simon Parsons for his brilliant teaching and for bringing to light, techniques that were applied to this project

REFERENCES

[1] Toussaint, M., Harmeling, S. and Storkey, A., 2006. Probabilistic inference for solving (PO) MDPs. Technical Report EDI-INF-RR-0934, School of Informatics, University of Edinburgh.

[2] Korf, R.E., 1988. Optimal path-finding algorithms. In Search in artificial intelligence (pp. 223-267). Springer, New York, NY.

[3] Lester, P., 2005. A* pathfinding for beginners. online]. GameDev WebSite. http://www. gamedev. net/reference/articles/article2003. asp (Acesso em 08/02/2009).

[4] Goyal, A., Mogha, P., Luthra, R. and Sangwan, N., 2014. Path finding: A* or dijkstra's?. International Journal in IT & Engineering, 2(1), pp.1-15.

[5] Qudrat-Ullah, H., Spector, J.M. and Davidsen, P. eds., 2007. Complex decision making: Theory and practice. Springer.

[6] Simon Parsons, 2022. Lecture 7: Complex decision making. https://uol.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=b69f615e-0f8f-45d0-9dcf-adef00962046

[7] Florescu, I., 2014. Probability and stochastic processes. John Wiley & Sons.

[8] Wikipedia, 2022. Markov decision process. https://en.wikipedia.org/wiki/Markov_decision_process

[9] Garcia, F. and Rachelson, E., 2013. Markov decision processes. Markov Decision Processes in Artificial Intelligence, pp.1-38.

[10] Peng, S., 1992. A generalized dynamic programming principle and Hamilton-Jacobi-Bellman equation. Stochastics: An International Journal of Probability and Stochastic Processes, 38(2), pp.119-134.

[11] Simon Parsons, 2022. Lecture 9: Markov Decision Process. https://uol.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=6f0e8ec4-045d-401d-b4c1-adf600963e25

[12] Chadès, I., Chapron, G., Cros, M.J., Garcia, F. and Sabbadin, R., 2014. MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. Ecography, 37(9), pp.916-920.

APPENDIX

Partial visibility set to TRUE					
average score					
scenario1		scenaio2		scenario3	
15by15	6.5	15by15	21.8	15by15	20.4
10by10	14.6	10by10	26.5	10by10	15.8
5by5	5.3	5by5	6.5	5by5	NA
average simulation time(seconds)					
scenario1		scenaio2		scenario3	
15by15	13.4	15by15	40.2	15by15	27.4
10by10	9.6	10by10	21.4	10by10	14.3
5by5	3	5by5	1.7	5by5	NA

Figure 6. This is an image of my stat table which shows Tallon's performance when partial visibility was set to TRUE for 3 scenarios that are outlines in more detail in the images below

Partial visibility set to FALSE					
average score					
scenario1		scenaio2		scenario3	
15by15	13.7	15by15	23.5	15by15	9.22222
10by10	15.6	10by10	16.9	10by10	19.7778
5by5	10.1	5by5	5.7778	5by5	NA
average simulation time(seconds)					
scenario1		scenaio2		scenario3	
15by15	16.2	15by15	23.6	15by15	11.6
10by10	11.2	10by10	7.6	10by10	10.3
5by5	4.7	5by5	3.1	5by5	NA

Scenario 1		
speed of meanies spawning	5	
number of bonuses	3	
number of pits	3	
Partial visibility	TRUE	

grid size 10 by 10

runs	simulation time length	Score	number of Meanies in view
1	4	2	1
2	4	12	1
3	16	28	3
4	7	13	1
5	21	30	2
6	3	11	1
7	10	5	2
8	8	14	2
9	5	22	1
10	18	9	1

grid size 5 by 5

runs	simulation time length	Score	number of Meanies in view
1	2	11	1
2	1	0	1
3	6	23	2
4	4	12	1
5	1	0	1
6	6	3	2
7	4	2	1
8	1	0	1
9	2	1	1
10	3	1	1

grid size 15 by 15

runs	simulation time length	Score	number of Meanies in view
1	8	4	1
2	9	4	1
3	17	8	2
4	14	7	0
5	6	3	1
6	13	6	1
7	12	6	1
8	14	7	1
9	29	14	1
10	12	6	1

Figure 8.

Scenario 2	
speed of meanies spawning	10
number of bonuses	6
number of pits	6
Partial visibility	TRUE

grid size 10 by 10

runs	simulation time length	Score	number of Meanies in view
1	5	2	1
2	2	1	1
3	11	35	2
4	10	45	2
5	22	41	3
6	40	50	3
7	36	28	4
8	16	18	2
9	17	18	2
10	55	27	2

grid size 5 by 5

runs	simulation time length	Score	number of Meanies in view
1	1	0	1
2	1	10	1
3	2	21	1
4	1	0	1
5	3	11	1
6	1	0	1
7	1	0	1
8	1	0	1
9	2	1	1
10	4	22	1

grid size 15 by 15

runs	simulation time length	Score	number of Meanies in view
1	7	3	0
2	15	17	1
3	32	16	1
4	6	3	1
5	11	5	2
6	19	9	1
7	26	13	1
8	163	91	2
9	112	56	1
10	11	5	2

Figure 9.

Scenario 3		
speed of meanies spawning	20	
number of bonuses	12	
number of pits	12	
Partial visibility	TRUE	

grid size 10 by 10

runs	simulation time length	Score	number of Meanies in view
1	3	11	0
2	5	22	1
3	65	52	3
4	2	1	0
5	5	2	1
6	28	24	1
7	16	8	1
8	5	32	1
9	5	2	1
10	9	4	1

grid size 15 by 15

runs	simulation time length	Score	number of Meanies in view
1	5	12	1
2	2	1	1
3	59	39	0
4	12	16	1
5	2	1	1
6	25	22	1
7	14	7	0
8	5	2	0
9	53	46	2
10	97	58	2

Figure 10.

Scenario 1		
speed of meanies spawning	5	
number of bonuses	3	
number of pits	3	
Partial visibility	FALSE	

grid size 10 by 10

runs	simulation time length	Score	number of Meanies in view
1	4	12	2
2	15	17	3
3	4	2	2
4	15	37	3
5	7	3	2
6	6	13	2
7	11	15	3
8	9	4	2
9	13	16	3
10	28	37	8

grid size 5 by 5

runs	simulation time length	Score	number of Meanies in view
1	4	12	2
2	5	22	2
3	3	11	1
4	4	12	2
5	3	1	1
6	2	11	1
7	6	3	2
8	5	12	1
9	9	14	2
10	6	3	2

grid size 15 by 15

runs	simulation time length	Score	number of Meanies in view
1	3	1	1
2	12	24	3
3	34	17	6
4	19	9	4
5	30	15	5
6	21	11	4
7	10	18	2
8	6	3	1
9	15	17	6
10	12	22	3

Figure 11.

Scenario 2		
speed of meanies spawning	10	
number of bonuses	6	
number of pits	6	
Partial visibility	FALSE	

grid size 10 by 10

runs	simulation time length	Score	number of Meanies in view
1	2	5	1
2	13	26	2
3	7	3	1
4	2	11	1
5	14	37	2
6	11	15	2
7	2	1	1
8	3	11	1
9	17	38	2
10	5	22	1

grid size 5 by 5

runs	simulation time length	Score	number of Meanies in view
1	1	0	1
2	2	21	1
3	1	0	1
4	8	14	1
5	1	0	1
6	4	2	1
7	7	13	1
8	2	1	1
9	2	1	1
10	3	11	1

grid size 15 by 15

runs	simulation time length	Score	number of Meanies in view
1	35	17	5
2	50	65	6
3	8	4	1
4	7	23	1
5	9	14	1
6	52	46	6
7	49	44	5
8	11	15	2
9	12	6	2
10	3	1	1

Figure 12.

Scenario 3		
speed of meanies spawning	20	
number of bonuses	12	
number of pits	12	
Partial visibility	FALSE	

grid size 10 by 10

runs	simulation time length	Score	number of Meanies in view
1	2	1	1
2	26	43	2
3	19	29	1
4	15	37	1
5	2	1	1
6	10	15	1
7	5	12	1
8	16	38	1
9	5	2	1
10	3	1	1

grid size 15 by 15

runs	simulation time length	Score	number of Meanies in view
1	12	6	1
2	17	18	1
3	3	1	1
4	6	3	1
5	4	2	1
6	10	5	1
7	9	14	1
8	14	7	1
9	34	27	2
10	7	3	1

Figure 13.

The Third scenario does not test a 5 by 5 grid for either partial or full visibility because it is not possible to simulate this as the grid dimensions are too small to spawn in all the assets

Image of rewards function which shows that although there were two bonuses, two meanies and three pits all active, the rewards table only has one present at each time step, the closest one to Tallon is the only one present.

This enables Tallon to get caught off guard by enemies or traps ahead of it when it is moving in the y axis.

```
Bonus closest to Tallon is:  (3, 4)
Meanie closest to Tallon is:  (2, 3)
Pit closest to Tallon is:  (2, 2)
Tallon is at:  [ 24 ] Bonueses unravelled:  [34, 44] Meanies unravelled:  [23, 14] Pits unravelled:  [13, 22, 11]
Policy:
 (1, 1, 2, 1, 1, 1, 1, 0, 2, 1, 1, 3, 0, 0, 1, 2, 2, 1, 2, 0, 2, 2, 2, 0, 0)
Rewards function:  [[-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-1.   -1.   -1.   -1.  ]
 [-1.   -1.   -1.   -1.  ]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [ 1.    1.    1.    1.  ]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]
 [-0.04 -0.04 -0.04 -0.04]]
At: 24  best move is going 0 north
```

Figure 14.