CMP9764M Comaring a Q-learning driven agent to an IRL based agent



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

[Introduction 2](#_Toc95791866)

[Concept 2](#_Toc95791867)

[Evaluation 3](#_Toc95791868)

[Appendix 5](#_Toc95791869)

*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

Preliminary results showed that the Q-learning is much faster than an IRL agent because of the need for human confirmation at each step as not only could the human also make mistakes at time since there is no policy being used aside from the human’s choice of what they think is the right action, the human can also carelessly classify bad actions as good and good as bad. Moreover the IRL agent needs to suggest a good alternative to an action if the human says an action is bad otherwise the IRL agent ends up stuck in its current position until a good move is suggested, this means the agent takes a lot more time to move if it keeps suggesting wrong actions, there needs to be a way to let the agent know an action is bad due. This can be done by looking at the last actions suggested by the agent and if they are bad actions then the agent suggests a new action at random that is not one of the previous actions. This reduces how long it takes the agent to move but also doesn’t take the freedom of choice from the agent because if it given exactly 3 chances to pick a right action then in a true deterministic world the agent would have picked a different potentially right action as there are only 4 actions.

When the Q-learning and IRL agent were tested for 40 iterations, the mean number of states the IRL agent took to get to the goal was 8.45, the maximum number was 46 and the minimum was 5. This showed that in most cases the IRL agent performed better. The maximum time taken to get to the goal only occurred on the first run of the experiment. This demonstrates that the agent leant the best actions to suggest as time went on.

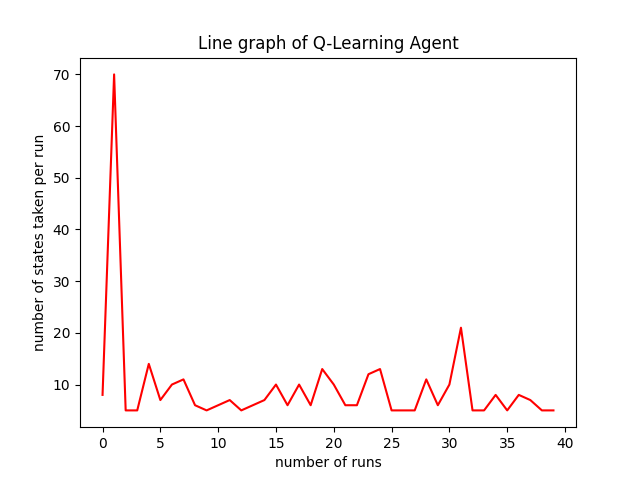


Figure 1 Plot of Q-Learning

The above figure shows that on the first run the agent took the most amount of steps to get to the goal but this reduced in the succeeding steps.

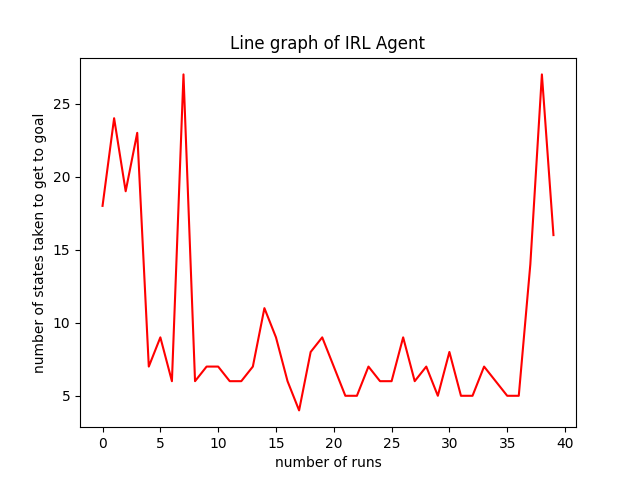
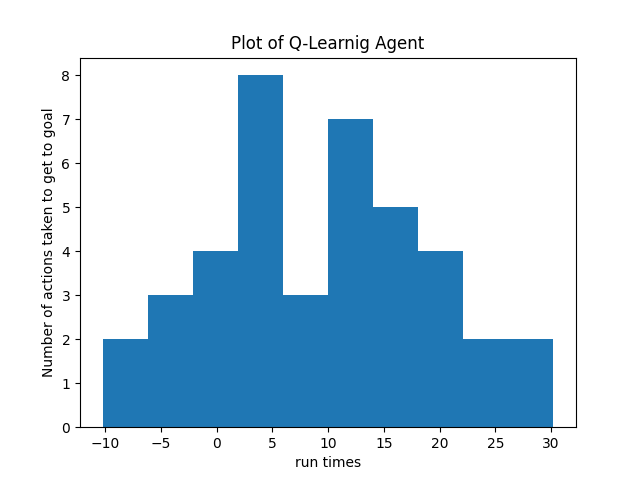
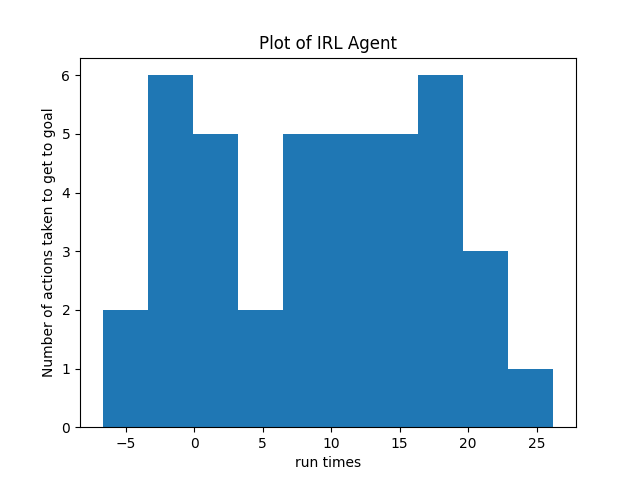


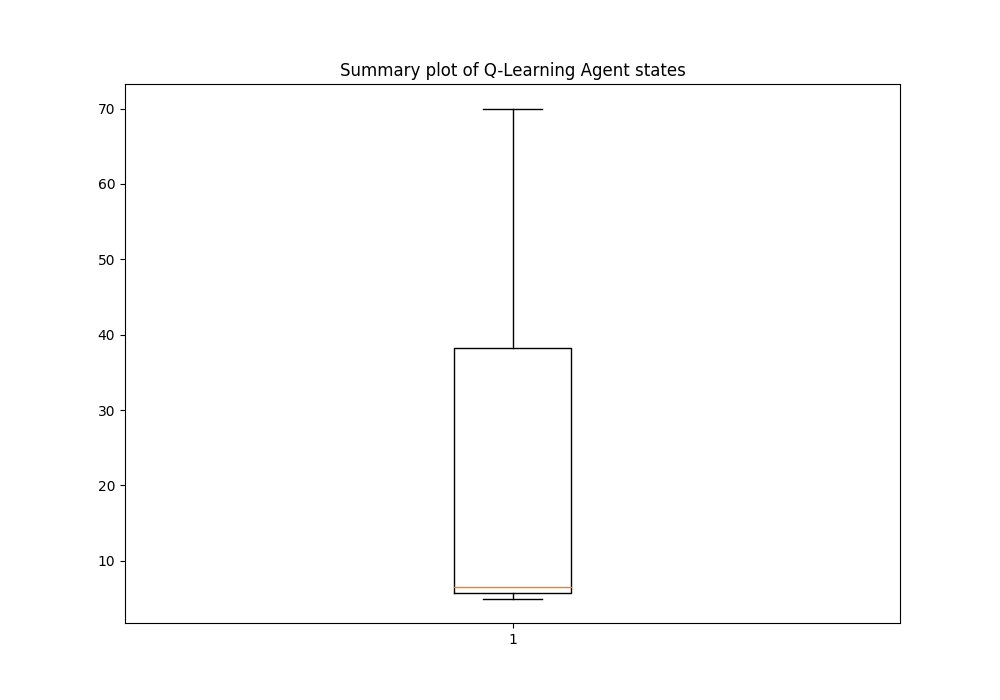
Figure 2 Plot of IRL Agent

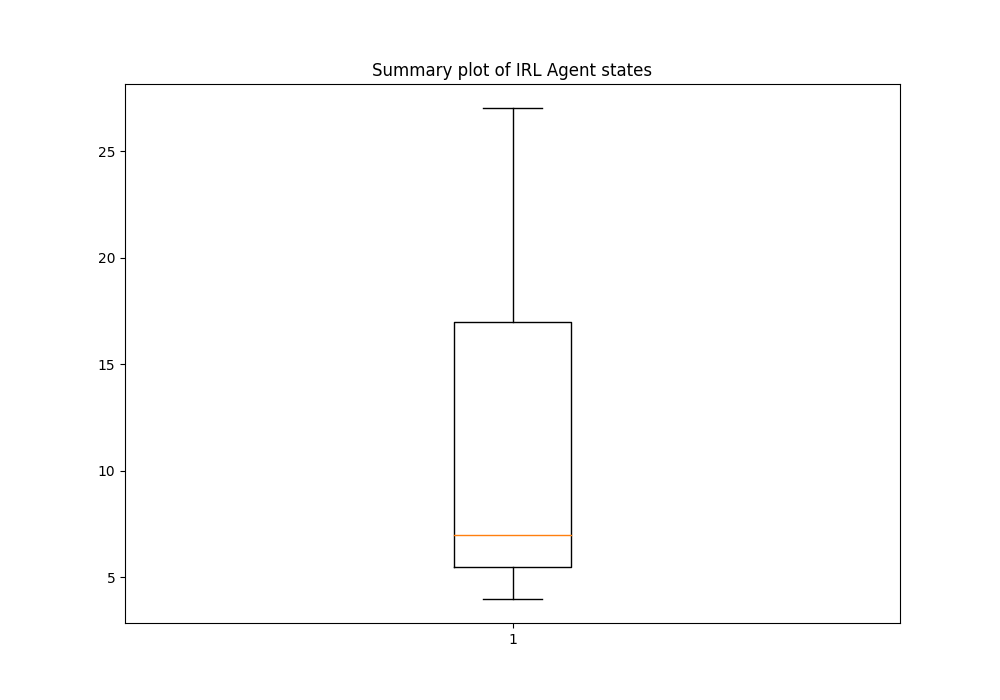
The above graph showed that the performance of the Interactive Agent Fluctuated and there was no clear increase in performance as the experiment carried on therefore although the agent made some improvements halfway through the run, this progress was lost on the last runs.



Above is a Histogram of the Performance of the Q-Learning Agent. The results show that out of the 40 runs the mean number of actions taken to get to the goal was 3 around 3.

The above plot for the IRL agent is a histogram demonstrating the distribution of actions taken in each run for 40 runs.

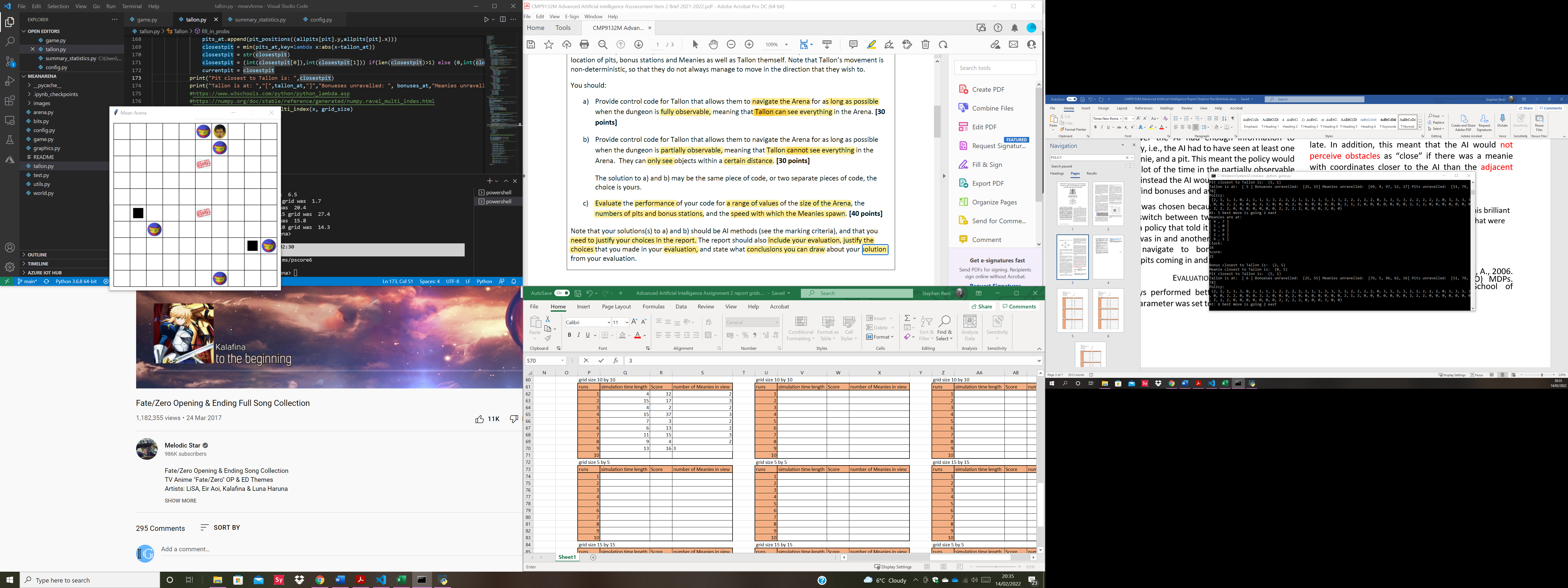


The above is a boxplot of the Maximum, minimum and median value of the number of states that the Q-learning Agent took over the 40 runs. The Median value is quite close to the minimum value which means that the maximum value was an outlier and resulting in the large contrast.

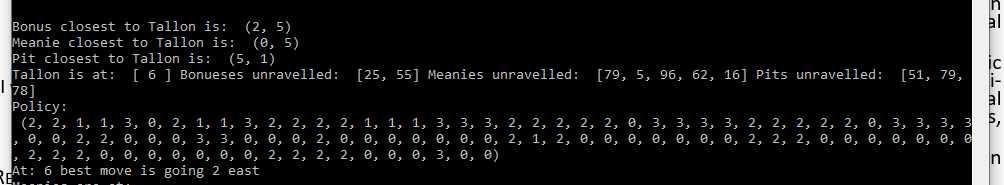
The above graph showed that the Median value was around the same as the median value of the Q-Learning Agent however the maximum number of states the IRL agent took across the 40 runs was significantly smaller than the maximum number of steps the Q-Learning Agent took at under 30 states. For the Q-Learning Agent however, this value was around 70. The upper quartile value of the IRL Agent was also lower between the ranges of 15 and 20 while the upper quartile of the Q-learning Agent was around 40 actions.

# Evaluation

The best policy seems to be..



1. 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.



1. 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.

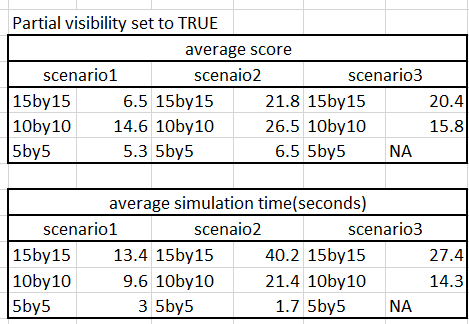
##### Acknowledgment

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

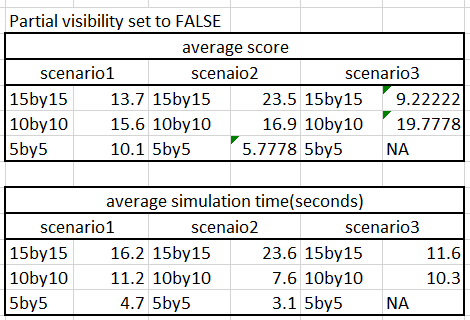
##### References

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.

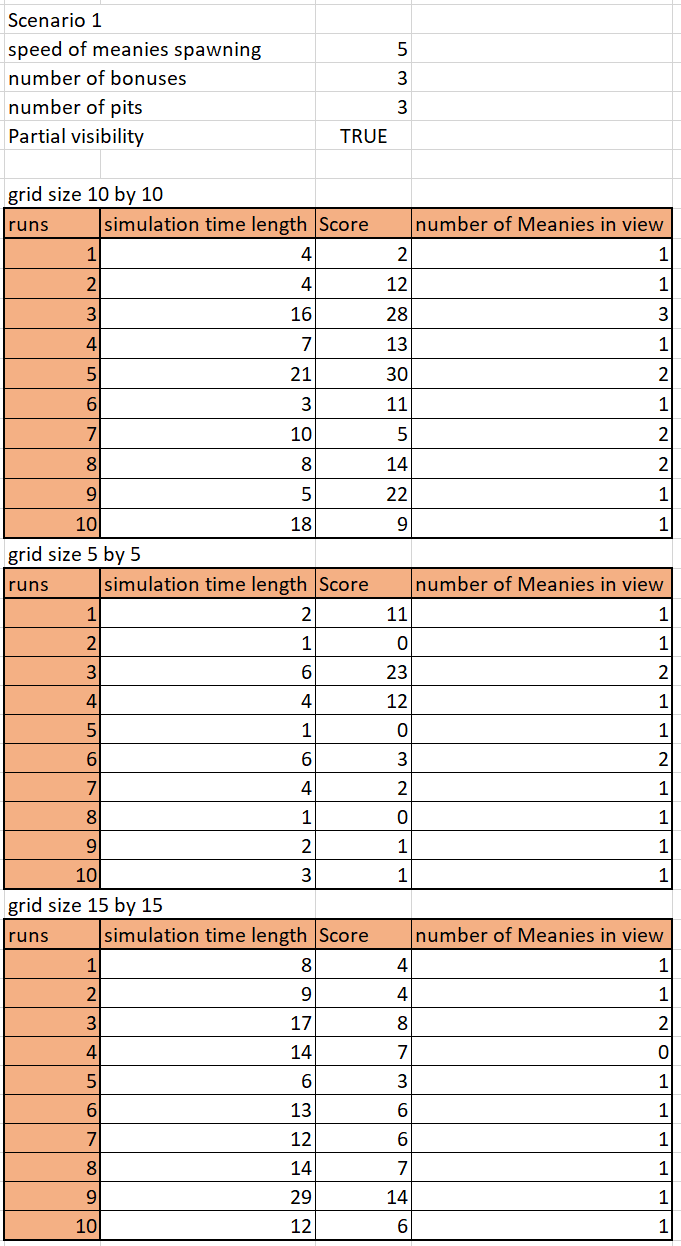
# Appendix

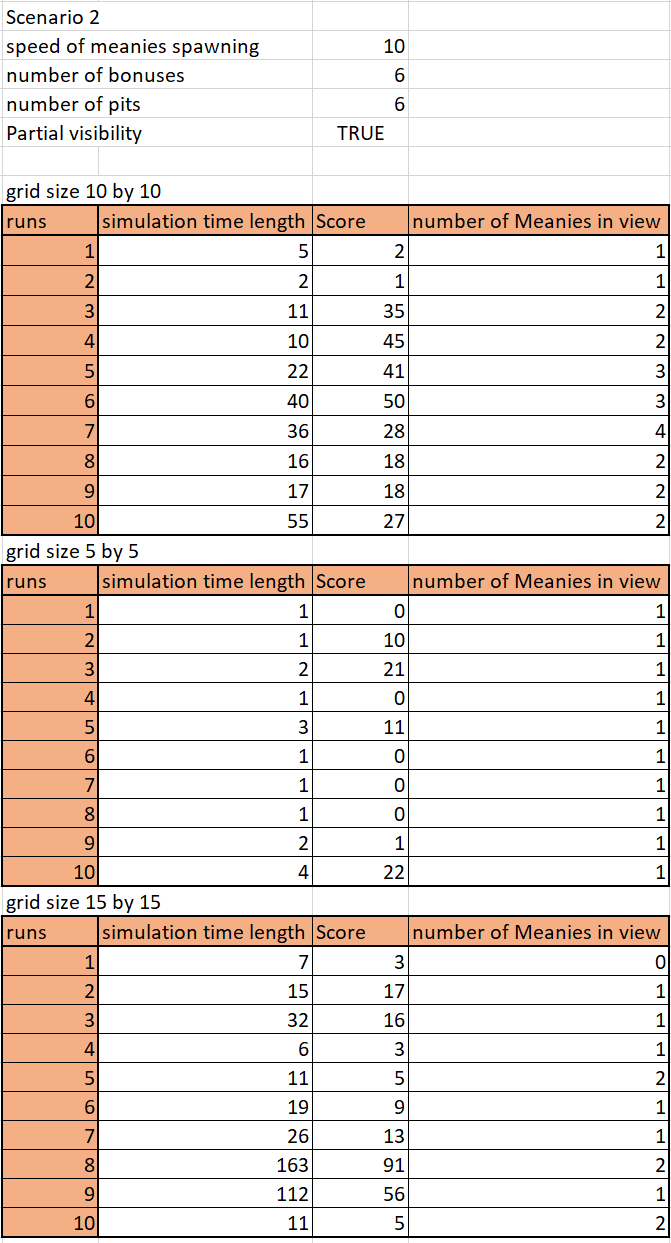
**

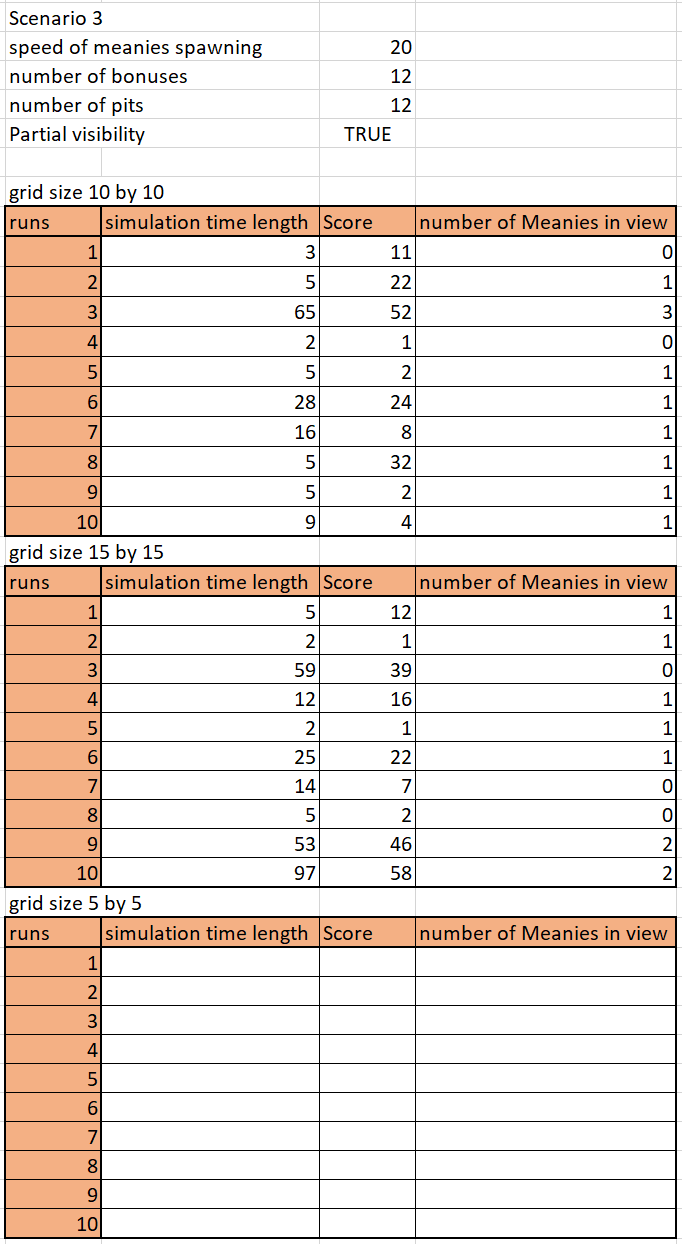
1. *This is an image of my stat table which shows Tallon’s performance when partial visibility was set to TRUE for 3 scenarios that are outlines in more detail in the images below*

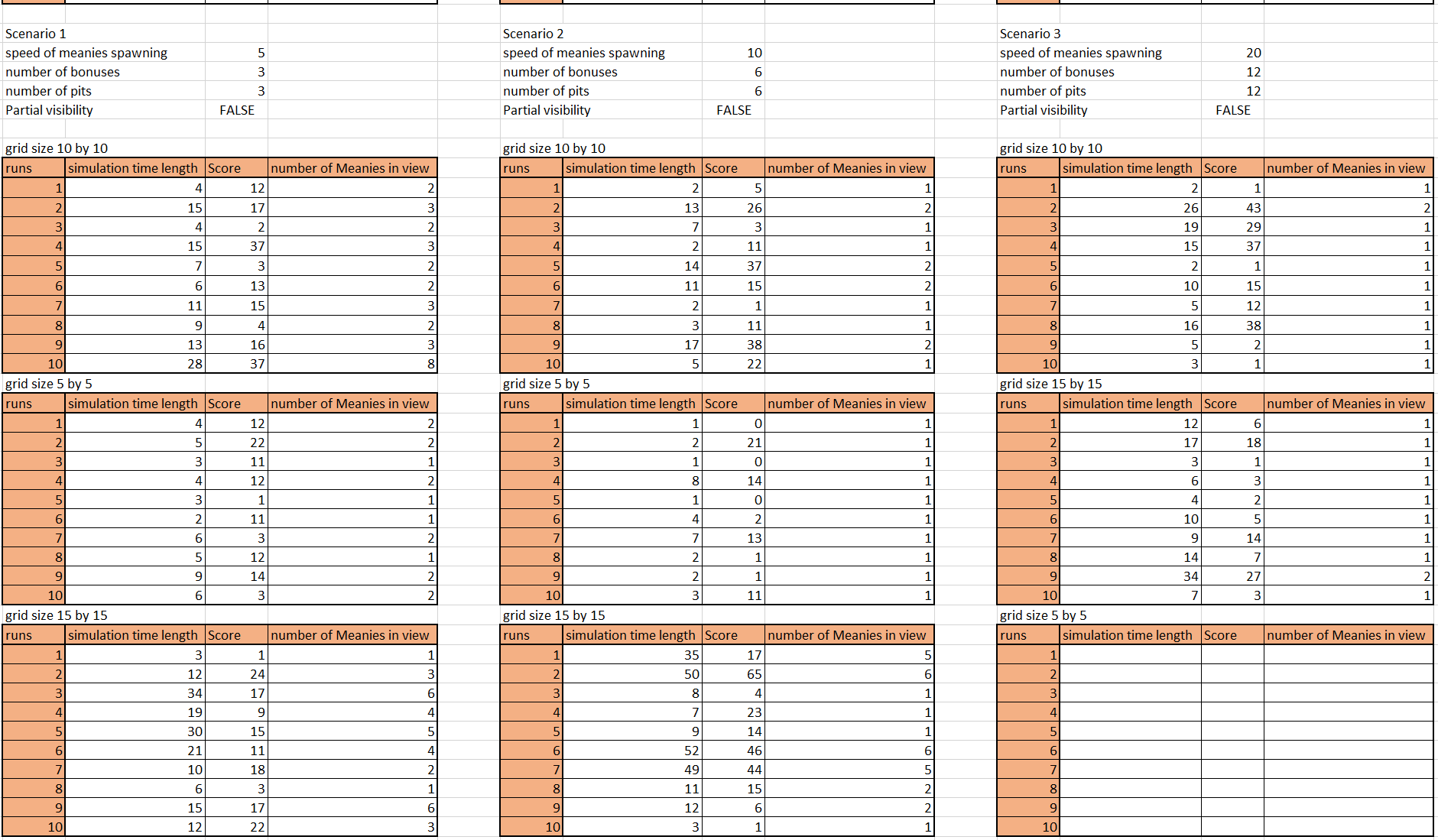
**

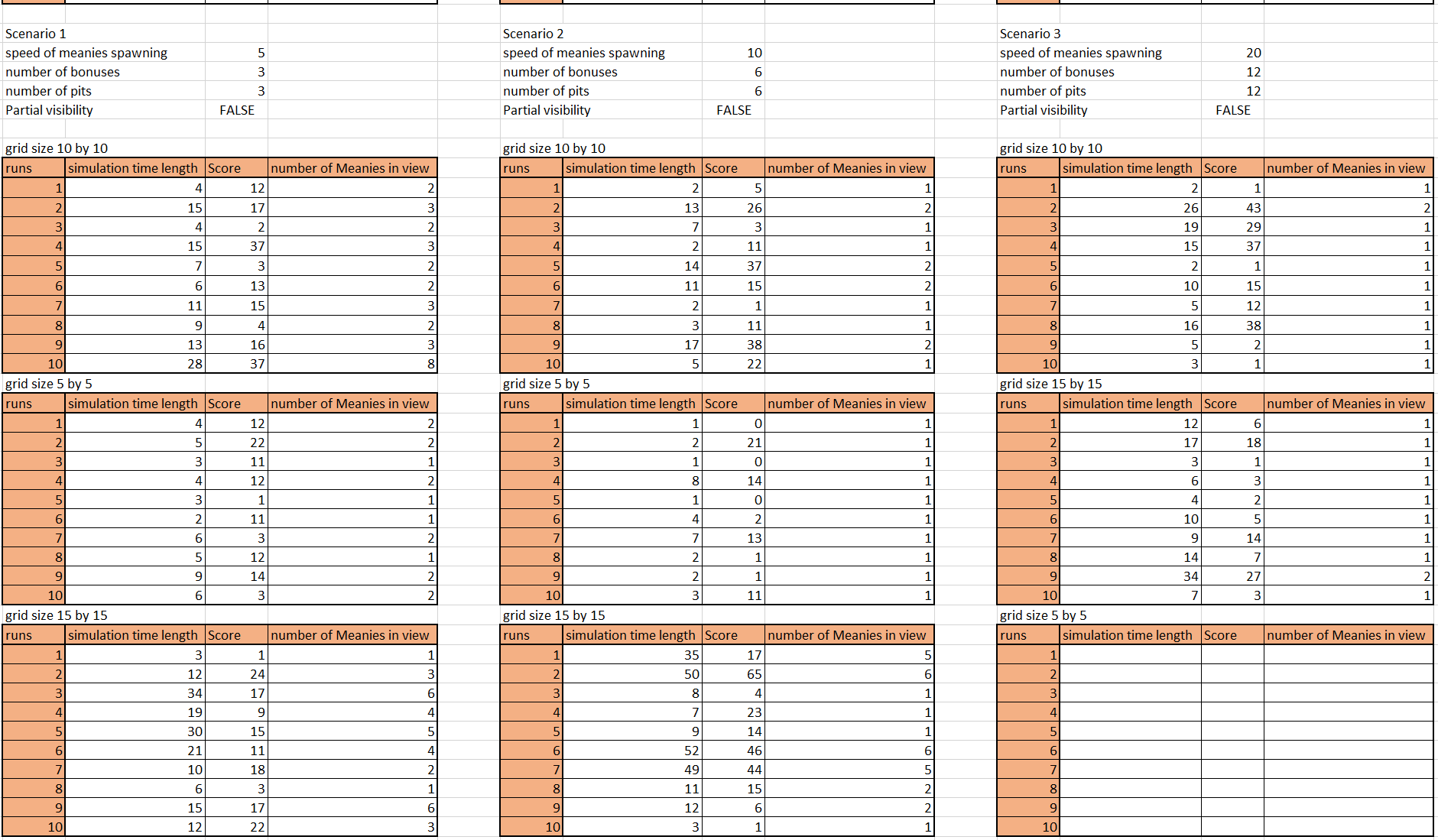
1. *This shows Tallon’s performance when partial visibility was set to FALSE for the 3 scenarios outlines in more detail in the images below*

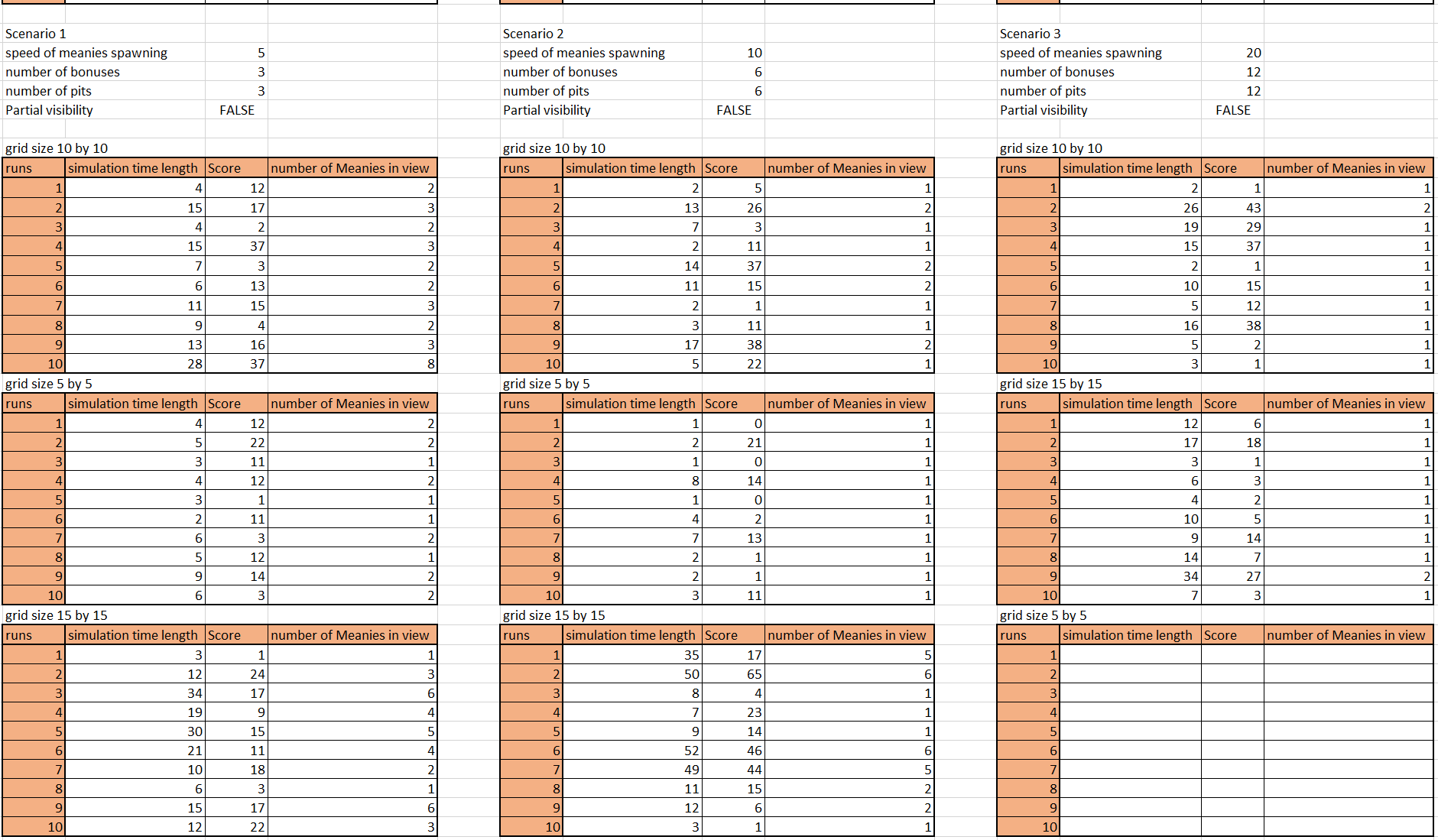
**

**

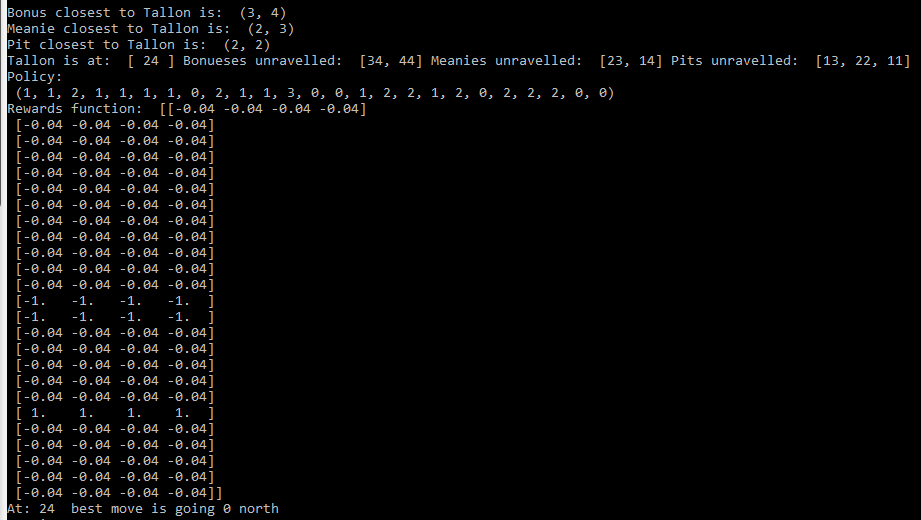
**

**

**

**

*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.*