Griffith School of Information Technology
Griffith University

3806ICT – Robotics, Agents and Reasoning
Trimester 1, 2021

# Grid World Maze Runner

Ryan Williamson, s5135470
Connor Forbes, s5068337

We, by signing this page, declare that the work presented in this report is all work done by us, unless appropriate reference has been made to the work of others. I acknowledge that should this not be the case the report will receive zero marks and due action may be taken.

**Submitted on:** ____________________

**Mark Received:** ____________________

# TABLE OF CONTENTS

## 1 Abstract

The following report focusses on model checking and reinforcement learning to gain knowledge and further insight into the area through research and experimentation. The domain of the testing is within a grid maze world and the problem given is to find the shortest most optimal path through the maze from the starting position to the goal position. For model solving, the Process Analysis Tooklit (PAT) and Reinforcement Learning (RL) were compared in terms of both quality of solution and time to solve the problem on a set of randomly generated mazes. To conduct the study, models in the domain were developed both within PAT and RL as well as a random maze generation program in order to ensure fair testing in all conditions. The results found the quality of solution to be comparable for both PAT and RL, however PAT had a significantly lower computation time.

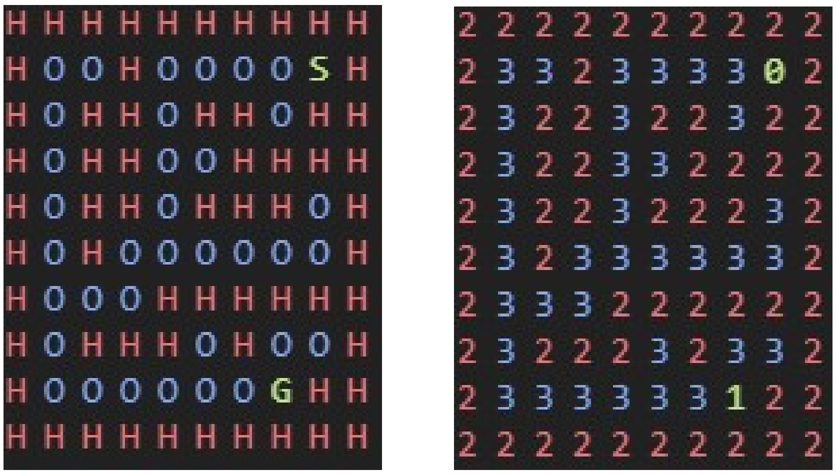
## 2 Model the Grid World in PAT

### 2.1 World Representation

Before the Grid World can be modelled with PAT each point needs to be parameterised as an integer as global constants can only represent boolean or integer values.This conversion is only required for PAT's internal computation and for readability convention will still be maintained. Each point within the grid world as been converted as per the following:

- **'O' for open space -** Modelled as a the integer 3
- **'H' for a wall -** Modelled as the integer 2
- **'G' for the goal location -** Modelled as the integer 1
- **'S' for the starting location -** Modelled as the integer 0

An example 10x10 grid world has been generated and can be seen below along with the PAT's internal representation of the world.



```
H H H H H H H H H H
H O O H O O O O S H
H O H H O H H O H H
H O H H O O H H H H
H O H H O H H H O H
H O H O O O O O O H
H O O O H H H H H H
H O H H H O H O O H
H O O O O O O G H H
H H H H H H H H H H
```

(a) Readable Grid World Representation

```
2 2 2 2 2 2 2 2 2 2
2 3 3 2 3 3 3 3 0 2
2 3 2 2 3 2 2 3 2 2
2 3 2 2 3 3 2 2 2 2
2 3 2 2 3 2 2 2 3 2
2 3 2 3 3 3 3 3 3 2
2 3 3 3 2 2 2 2 2 2
2 3 2 2 2 3 2 3 3 2
2 3 3 3 3 3 3 1 2 2
2 2 2 2 2 2 2 2 2 2
```

(b) Internal PAT Representation

Figure 1: 5x5 Grid World Representation

### 2.2 Game Choices

Four guarded choices are available at each game state that can be taken to move the current position around the maze, the guarded process serves to prevent any illegitimate states from being reached where the position is outside the bounds of the maze.

- **Move Up -** Move the current location up one row
- **Move Down -** Move the current location down one row
- **Move Left -** Move the current location left one column
- **Move Right -** Move the current location right one column

### 2.3 Assertion

With the Game choices and actions modelled, an assertion can be made that the goal state is reachable for the current board. The goal state can be defined as

$$Goal = Board[Row][Col] == G \tag{1}$$

Where Row and Col are the current position, meaning that the state has traversed through the maze from the starting point to the end location.

### 2.4 Model Verification Testing

Using PAT's inbuilt model verification tool and the Game Choice, World Representation and Assertion defined the model checker can run to determine if the current maze is indeed solvable and the resultant optimal path. The below optimal path has been given for the grid world defined within Figure 1, the time taken to calculate the path was 0.0004949s

$$go\_left- > go\_left- > go\_left- > go\_left- > go\_down- > go\_down- > go\_down- > go\_down- > go\_left- > go\_down- > go\_left- > go\_left- > go\_down- > go\_down- > go\_right- > go\_right- > go\_right- > go\_right- > go\_right- > go\_right$$

## 3 Model the Grid World Maze for RL

Reinforcement Learning is an unsupervised learning method where some states, transistions and actions have a corrosponding rewards of actions and then given a goal will determine the method to maximise the reward. Each step the algorithm takes amounts to -1 point, while running into a wall is -5 points. Reaching the goal awards the agent with 30 points. There are many different variations of the reinforcement learning algorithm each with their own advantages and disadvantages, for this task a Q-learning algorithm will be used. Q-learning attempts to establish an action-value function, which is the expected value of taking an action at a given state. The training phase attemps to construct a Q table that can be used during test time to take the optimal path to reach the goal. The agent attemtps to maximize its score and the best way to do that is reaching the goal in the minimum number of steps.
The algorithm involves several tuning parameters that an drastically alter the performance of the agent, the learning parameters for this algorithm were tested on various maze sizes and these parameters were selected and maintained for all maze sizes to ensure a fair test:

- **Gamma: 0.99 -** The discount rate of the algorithm
- **Alpha: 0.1 -** The learning rate of the algorithm
- **Epsilon: 0.2 -** The probability that the algorithm will take an exploratory action during training
- **Maxiter: 1000 -** The maximum number of iterations for the algorithm to take
- **Maxsteps: 1000 -** The maximum number of steps that the algorithm can take before termination

The model is broken up into two seperate phases, first the training phase where the tuning parameters effect how many iterations it runs for and how many steps allowed within each iteration. The actions of the agent at each step is then effected by the Gamma, Alpha and Epsilon value to determine how it explores the maze.During the training phase the model populates the Q table, which represents the action-value function that is used during testing. At each state it references the table and takes the best action possible to reach the goal state, this process is quick and results in extremely quick testing time regardless of the maze size.

## 4 Parameterise the Grid World

For the maze generation, a variation of the Randomized Prim's Algorithm[1] was implemented to generate the maze. The algorithm works as follows:

1. Start with a NxN maze full of univisited points (U)
2. Pick a random cell, mark it as an open space (O) and mark the four directly adjacent (top, left, right, bottom) cells as walls (H) and add to the list of walls
3. While there are walls in the list:
    (a) Randomly pick a wall from the list and check how many adjacent cells are open spaces (O). If only one of the four surrounding cells is an open space then:
        i. Mark the wall as an open space (O) and remove from the list of walls
        ii. Mark the surrounding unvisited cells (U) as walls (H) and add to the list of walls
    (b) Else:
        i. Remove the wall from the list of walls
4. Once iteration is complete, mark the remaining unvisited cells as walls
5. Add a starting point (S) randomly at either the top left or top right open space (O) in the maze
6. Add the goal point (G) randomly at either the bottom left or bottom right open space (O) in the maze

The above algorithm will always result in a closed maze which contains an open path from the starting point (S) to the goal (G). Any size maze can be generated by providing the width and height as arguments when running the file from the command line (e.g. 'python ./grid-world-generation.py 15 15' for a 15 by 15 maze). Examples of mazes generated by the code can be found in Appendix A.

## 5 Translate the Grid World to a PAT Model and RL Model

Slight modifcations are required to convert the generated maze world from the generated output to eligible PAT and RL models. Extending on the end of the maze generation program two text files are created within the Mazes subdirectory, first the PAT model is written in the following format:

- Define the starting X position
- Define the starting Y position
- Define the width of the grid world
- Define the height of the grid world
- Define the generated maze as a 1D array

These items are exported to a .csp file that can be then imported to the PAT model checker for verification, these parameters must also be removed from the orginal model to prevent overwriting the variables. Following from this the RL model is then written to a .txt file, the RL model only requires the generated maze world as a one dimensional array with no other defining variables. No modification is requried as the initial model as RL was designed to read the flattened maze array from a text file.

## 6 PAT vs RL for Maze Runner

An experiment has been conducted to compare PAT model verification vs Reinforcement Learning in the domain of the maze runner. The experiment will be conducted fairly by testing the models on randomly generated mazes of varying size, starting from 5x5 up to 200x200. Identical grids will be used for comparison with the times for each model to run. The following parameters will be used for comparison:

- **Time Taken -** How long for the model to determine the optimal path
- **Length of Path -** How long the optimal path is

In order to form a fair comparison the training time required for reinforcement learning will also be measured and factored in as it is a requirement prior to the model running. As reinforcement learning's performance can also be heavily modifed by the identified tuning parameters these will also remain constant throughout experimentation, the same values identified within Section3 will be used. The experiment will be completed across seven different generated mazes, beginning with smaller mazes and then spiradically increasing to get a broad overview how each model works as the complexity significantly increases. A visualisation of the generated mazes has been attached within Appendix A. The table below shows the collected results from the experimentation:

| Maze Size | Time Taken (s) | Optimal Path Length | Training Time (s) | Total Time |
|---|---|---|---|---|
| 5x5 | 0.0000 | 6 | 0.21 | 0.21 |
| 10x10 | 0.0010 | 14 | 0.52 | 0.521 |
| 20x20 | 0.0010 | 33 | 2.19 | 2.191 |
| 40x40 | 0.0010 | 76 | 8.51 | 8.511 |
| 60x60 | 0.0050 | 177 | 87.38* | 87.385 |
| 100x100 | 0.0230 | Did not complete | 280.60* | 280.60 |
| 200x200 | 0.0230 | Did not complete | 396.48* | 396.482 |

Table 1: Reinforcement Learning Results

| Maze Size | Time Taken (s) | Optimal Path Length |
|---|---|---|
| 5x5 | 0.0098863 | 6 |
| 10x10 | 0.0007578 | 14 |
| 20x20 | 0.0049444 | 33 |
| 40x40 | 0.0500344 | 76 |
| 60x60 | 0.1773686 | 177 |
| 100x100 | 0.7564199 | 228 |
| 200x200 | 8.874657 | 476 |

Table 2: PAT Verfication Results

As can be seen, the PAT model verification performs significantly better in all aspects of testing. Without modfiication to the tuning parameters for larger grid sizes reinforcement learning begins to struggle immensely with exploring enough of the state space to reach the goal location and properly populate enough Q values. With the drastic increases in training time as the grid sizes increases reinforcement learning becomes extremely undesirable for model verification. A time comparison for the execution time has been visualised and can be seen within Appendix D, it is evident that the total runtime of reinforcement learning grows at an exponential rate compared to the relatively linear growth of PAT. In order to determine what sort of further tuning would be required to the model, the parameters were altered for the 60x60, 100x100 and 200x200 mazes to reflect:

- **Maxiter: 3000 -** The maximum number of iterations for the algorithm to take
- **Maxsteps: 5000 -** The maximum number of steps that the algorithm can take before termination

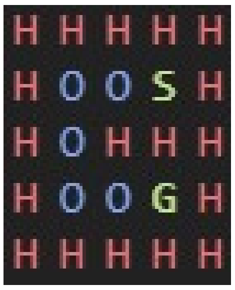
The adjusted parameters can be seen with the table noted by *.This was done to evauluate if the model required further training in order to be able to solve find the goal, as can be seen by the figures within Appendix C prior to the adjusted parameters during the testing phase the model was unable to find the goal state. However, after the max iterations were raised to 3000 and the maximum number of steps was raised to 5000, it was able to find the goal and the model was solvable. This is because the model has more steps to reach the goal, while also allowing enough iterations for the model to converge. This indicates that the larger 100x100 and 200x200 models are also solvable however will require even further tuning and more likely more training time.

To ensure a fair test, all testing was completed on the same hardware and software versions, these can be seen within Appendix B.

## 7 REFERENCES

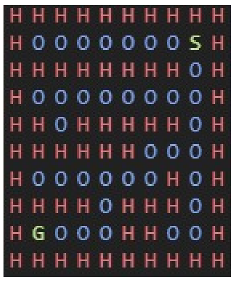
[1] A. Kozlova, J. A. Brown and E. Reading, ”Examination of representational expression in maze generation algorithms,” *2015 IEEE Conference on Computational Intelligence and Games (CIG),* 2015, pp. 532-533, doi: 10.1109/CIG.2015.7317902.

## A Appendix: Tested Maze Visualisations



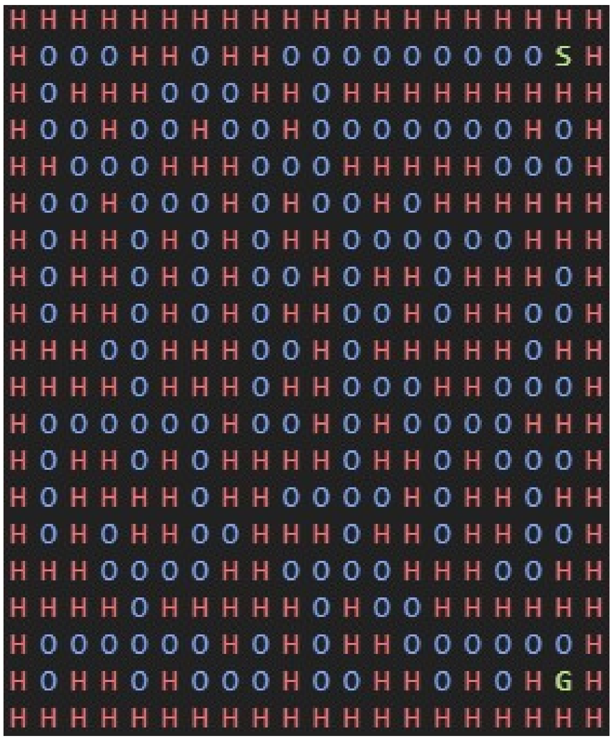
```
H H H H H
H O O S H
H O H H H
H O O G H
H H H H H
```

Figure 2: 5x5 Maze



```
H H H H H H H H H H
H O O O O O O O S H
H H H H H H H H O H
H O O O O O O O O H
H H O H H H H H O H
H H H H H H O O O H
H O O O O O O H O H
H H H H O H H H O H
H G O O O H H O O H
H H H H H H H H H H
```

Figure 3: 10x10 Maze



```
H H H H H H H H H H H H H H H H H H H H
H O O O H H O H H O O O O O O O O O S H
H O H H H O O O H H O H H H H H H H H H
H O O H O O H O O H O O O O O O O H O H
H H O O O H H H O O O H H H H H O O O H
H O O H O O O H O H O O H O H H H H H H
H O H H O H O H O H H O O O O O O H H H
H O H H O H O H O O H O H H O H H H O H
H O H H O H O H O H H O O H O H H O O H
H H H O O H H H O O H O H H H H H O H H
H H H H O H H H O H H O O O H H O O O H
H O O O O O O H O O H O H O O O O H H H
H O H H O H O H H H H O H H O H O O O H
H O H H H H O H H O O O O H O H H O H H
H O H O H H O O H H H O H H O H H O O H
H H H O O O O H H O O O O H H H O O H H
H H H H O H H H H H O H O O H H H H H H
H O O O O O O H O H O H H O O O O O O H
H O H H O H O O O H O O H H O H O H G H
H H H H H H H H H H H H H H H H H H H H
```

Figure 4: 20x20 Maze

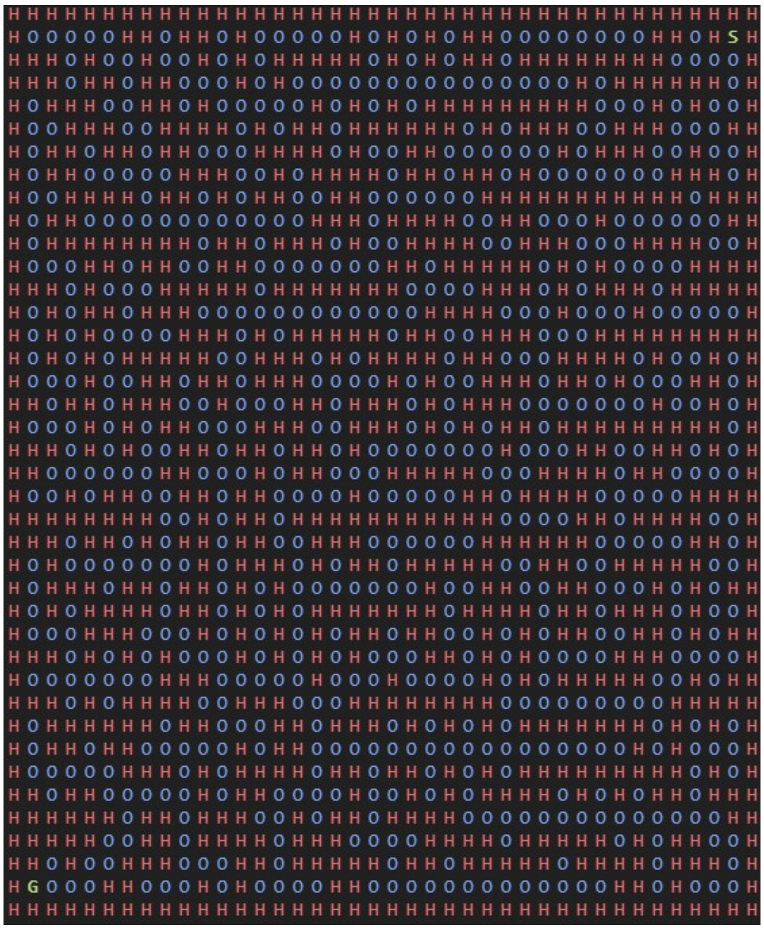


Figure 5: 40x40 Maze

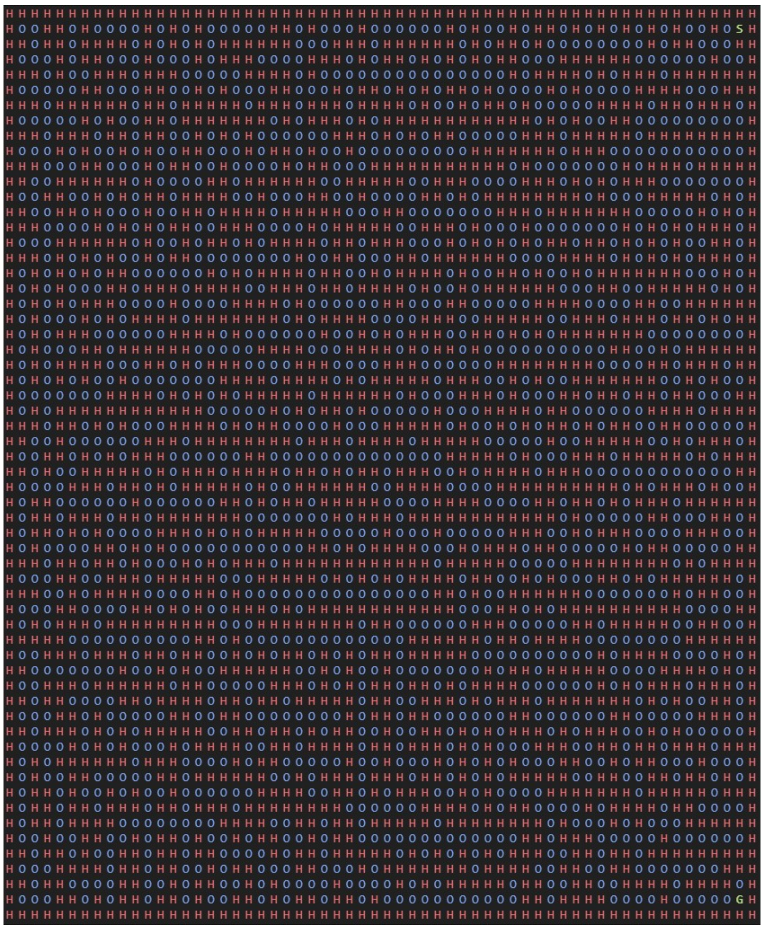


Figure 6: 60x60 Maze

## B Appendix: Testing Environment

The tested programs were run on the following hardware:

- **Processor:** AMD Ryzen 5 5600X
- **GPU:** NVIDIA RTX 3070
- **RAM:** 32GB 3200MHz DDR4

And the following software:

- **Python Version:** 3.8.10
- **Windows Version:** 10.0.19043
- **PAT Version:** 3.5.1

## C Appendix: Adjusted Tuning Parameters

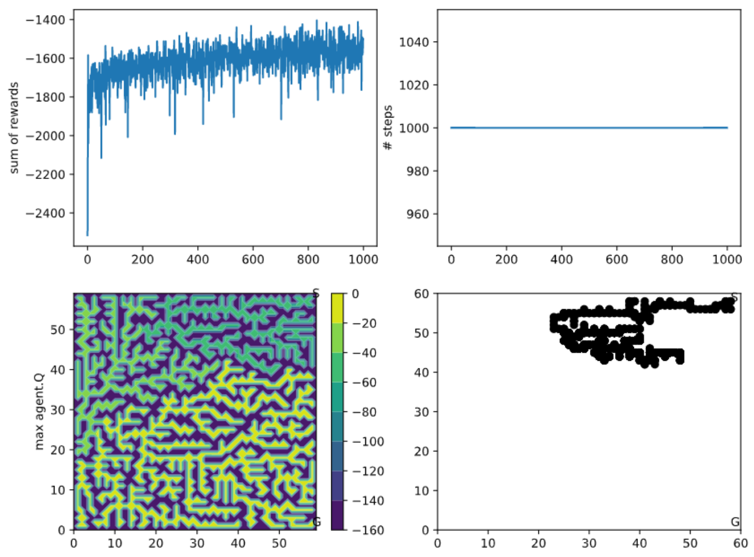


Figure 7: Original Tuning Parameters

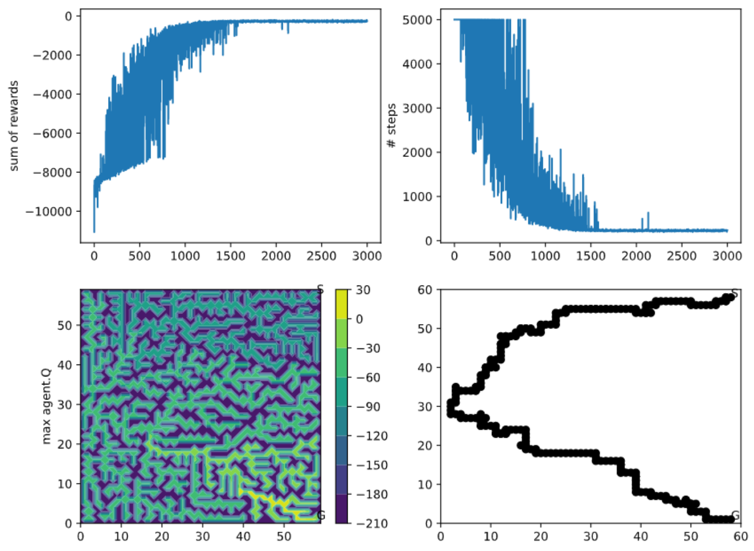


Figure 8: Adjusted Training Parameters

## D Appendix: PAT vs RL Execution Time

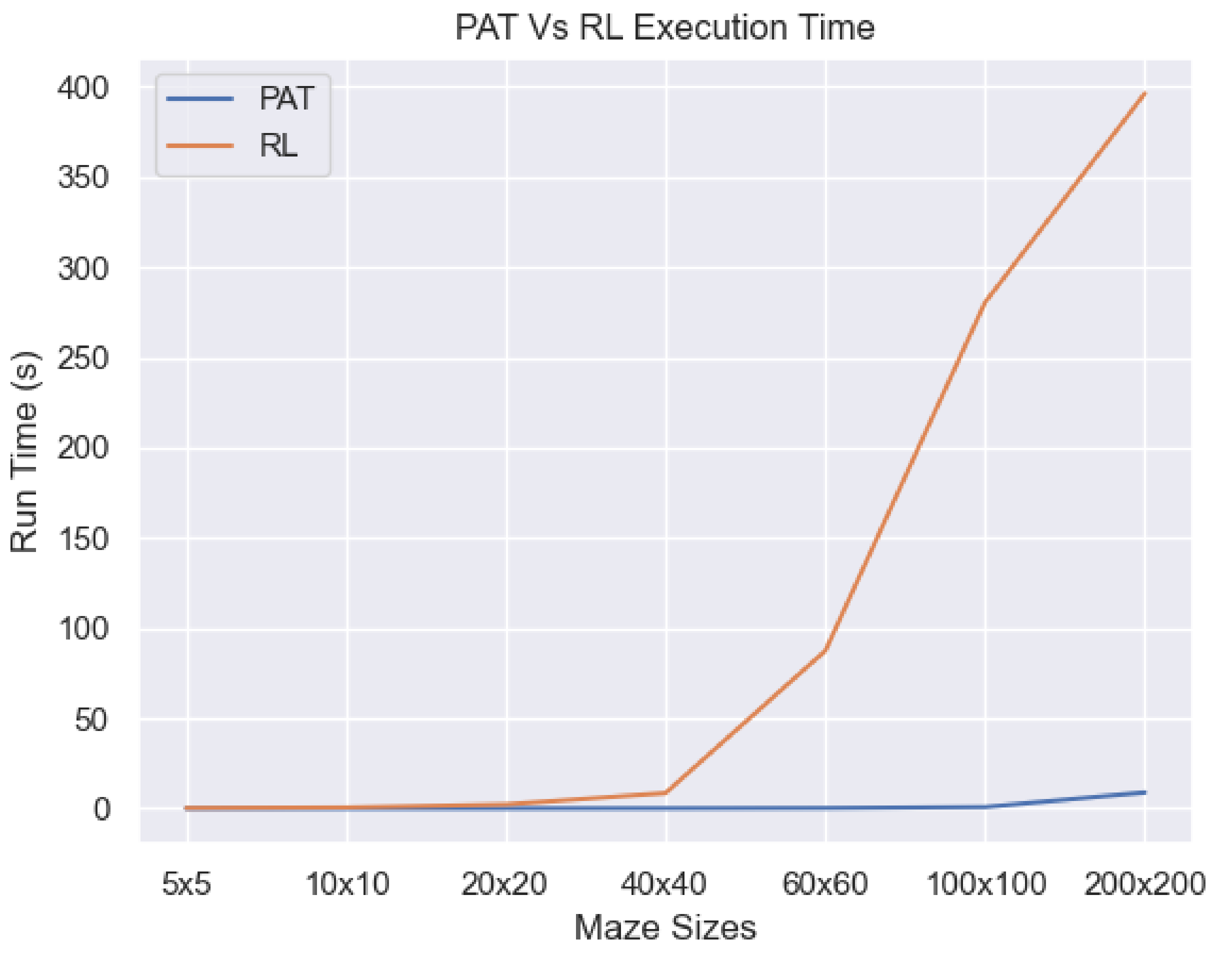


Figure 9: PAT vs RL Execution Time