Machine Learning Engineer Nanodegree

Capstone project: Micromouse project simulation

Kenneth Mora
July 16th, 2017

Definition

Project Overview

This project is inspirated in Micromouse competition that it is an event where a robot mouse try to reach the center of a maze starting from a corner in an optimal way. This project is about a simulation following tha same idea in a virtual maze.

Problem Statement

This problem is about a robot mouse navigating a maze with the goal to reach the center following the best path. This is a kind of reinforcement learning problem. The first run the mouse explores the maze in a maximum of one thousand steps and to do a good exploration is necessary to implement techniques to avoid to navigate the same area many times or get stuck in a dead end path. Always the robot starts from the corner. In the next run it tries to find the shortest path to reach the goal in the center in the less number of steps using the information learned in the first run for this it was used the a* algorithm. This algorithm gets the best results when the exploration in run 1 was broad.

Metrics

The robot's score for the maze is equal to the number of moves required to execute the second run, plus one thirtieth the number of moves required to execute the first run. A maximum of one thousand time steps are allotted to complete both runs for a single maze. The best score would be to reach the goal in the minimum number of moves in both runs therefore it performing little exploration but taking the short path.
Anothers extra metrics was added as the coverage (percentage of maze location explored) to check if high exploration give better results geting the optimal path in run 2. Other metric added was path length (number of steps) because a move can take 0-3 steps in the maze and maybe it's possible to find differents paths with the same number of moves and different path length.

Analysis

Data Exploration

The initial position for the virtual robot is the left bottom corner. The goal are four squares in the center. There are 3 mazes with dimensions 12x12, 14x14 and 16x16.
The virtual mouse has three sensors indicating the number of step available without pass by wall: forward, left and right. Each turn the robot can move a maximum of three steps and rotate 0, 90 or -90 degree. In this case the robot can't move backward receiving a negative step, all displacement are positive (forward) if the robot wants return must turn 90 degree twice.

Exploratory Visualization

It used the file showmaze.py to see the mazes using the next command: **python showmaze.py <*maze name*>**

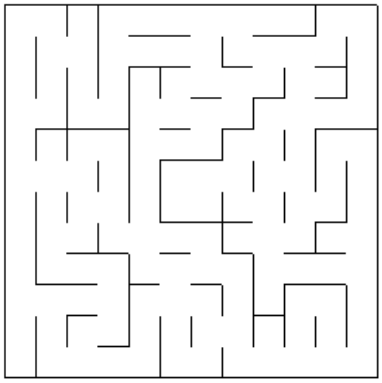

Fig. 1: This is the maze 1 with size 12x12

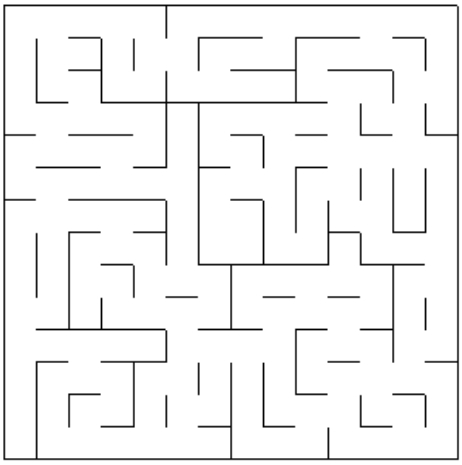

Fig. 2: This is the maze 2 with size 14x14

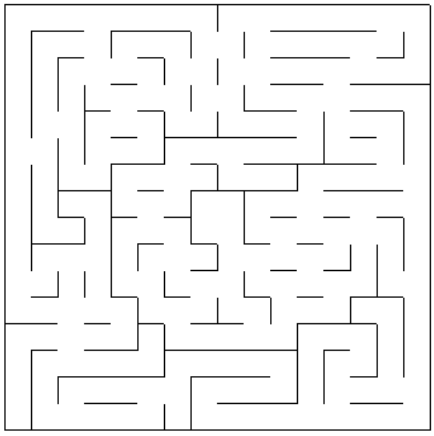

Fig. 3: This is the maze 3 with size 16x16

It detected the following dead end path in the mazes visually.

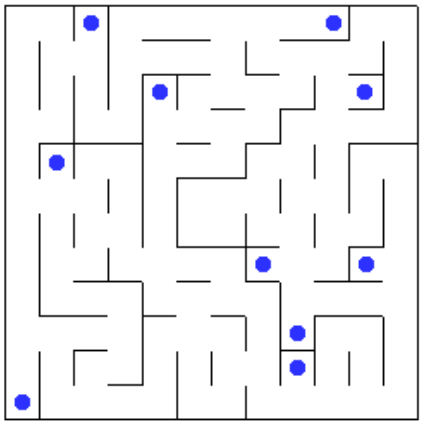

Fig. 4: Dead end paths in maze 1

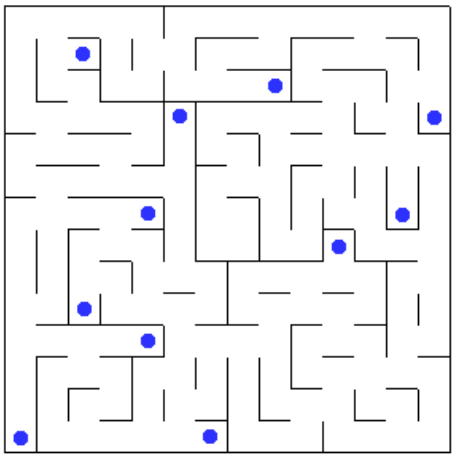

Fig. 5: Dead end paths in maze 2

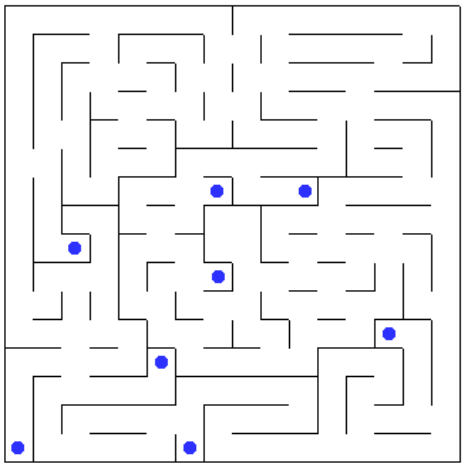

Fig. 6: Dead end paths in maze 3

It's important to avoid these locations because the robot gets stuck there and spends at least two moves trying to leave of the place (rotating twice 90 degrees). The second time there doesn't contribute anything to the exploration.

Algorithms and Techniques

In the first run it's necessary to explore the maze to know where there are walls. It used the following techniques:

- Simple random move.
- Dead end path detection: dead end paths are detected when the virtual robot enters in one of them is saved the location and then the next time the robot detects it as a possible move is removed as a choice.
- Counting number of visited locations: the robot reminds the number of times that it visited a location and when it tries to move to the next one get the locations less visited and chooses one.
- Calculating heuristic values: it calculates the distance from the current location to the goal in each turn and when the robot tries to move the next time then it chooses the lower value.

In the second run, the robot already explored the maze and reached the goal then it used the a* algorithm to follow the best path known from the start position to the goal. The a* algorithm calculates distance (in this case Manhattan distance) from the current position to the goal and then it gets the neighbors and it chooses one with the lower movement cost (in this case 1 for moving towards a known neighbor or 100 otherwise) and so on until the goal. Unexplored area has a high cost and the neighbors between walls are ignored for that reason the a* algorithm doesn't take those paths. Here there is a great visualization in how the standard algorithm works.
Sometimes a* algorithm doesn't get the optimal path because the robot when reached the goal the first time, it finished the run and it didn't explore all the maze locations.

Benchmark

The benchmark is to reach the goal in less than 1000 moves in both runs for each maze.

Methodology

Data Preprocessing

For running the main program it added two new parameters to the original code. One of them is called smart level to know what improvement is used in that execution and the other is a boolean value to know if the robot when reachs the goal will continue exploring or not.

Implementation

The starter code had functions made to load maze from files, read its structure, visualize it and it had a sensor validation of the possible direction and rotation that the mouse could take.
It added a constraint in the movement of the robot for only go forward because it's weird to see a mouse walking backward, if it wants back must turn 90 degree twice. If the robot get in a dead end path automatically it rotates in the next move because sensors don't detect possible moves.
It worked the basic random move for the virtual robot in the first run and a method to detect the center so to know when it arrived to the goal. Then it created a method to store a list of walls detected with the robot sensor in every move.
It added states for the robot to know in which stage is found. The first state is 'run 1' and it's set when the robot is created. When the robot reachs the goal pass to 'exploring' state. This state is jumped if exist a flag indicating there is no exploration stage otherwise this will end when the maximum number of moves is reached (1000) or all locations are visited. After that, the next state is 'run 2 and the last one is 'ended' when the goal is reached in the second run. There is one more state called 'goal unreachable' and this is set when the robot reachs the maximum number of moves in run 1 and didn't reach the goal. This avoid to execute moves in run 2 thus the robot stays static.
After that it was necessary to develop the A* algorithm in the second run so it's possible to get the firsts complete scores of a solution. It took the base algorithm from the Red Blob Games page[1]. This algorithm was modified to remove the nodes reached through a wall if it was in the list of walls detected when it searched for neighbors. Also it was modified to add more neighbors to three steps from the start location. A penalty was added in run 2 when the location wasn't visited to avoid follow an unknown path.
Then the idea was to improve the score exploring better and efficient in the first run. Three improvements were implemented: a dead end path detector, the counting of locations visited and adding heuristic values to the goal. It added a variable called smart_level to know which of these improvements was applied in the run.
In the improvement for the dead end path detection it added a list storing locations with this problem and this locations were removed of the options to choose for the next robot move. This list was added to the output file.
In the second improvement it added a dictionary with locations containing the number of visits and every move the robot chooses the location with lower visits.
In the last improvement it executed the a* algorithm in every move to calculate the distance from the current position to the goal. The heuristic used is the Manhattan distance.
The last implementation was an output file with results to see how improved the run with the techniques and algorithms and then to compare between them.
It was a challenge to adapt the code found of the a* algorithm to the starter code provided for the micromouse project because the algoritm managed the walls like nodes and it was necessary to manage it like edge between nodes. Other difficulty was to visualize the results and know what happened in every step with the techniques proposed because sometimes it gave weird results for that reason it

created an output file with interesting data about the run.

Refinement

In the first run, the techniques proposed (dead end path detection, counting of locations visited and adding heuristic values to the goal) will improve the basic random move created as a first approach.
The a* search was improved when it removed a condition to finish prematurely after the goal was reached because now it explores all the maze.

Results

Model Evaluation and Validation

The optimal path is gotten with the a* algorithm and exploring completely the maze. These are the results in all mazes:

Maze	Path length	Number of moves
1	33	17
2	43	22
3	52	25

In the first maze the robot follow this locations as the optimal path:
(0, 0), (0, 2), (1, 2), (1, 0), (4, 0), (4, 2), (5, 2), (5, 3), (7, 3), (7, 0), (8, 0), (11, 0), (11, 3), (8, 3), (8, 5), (7, 5), (7, 6), (5, 6)

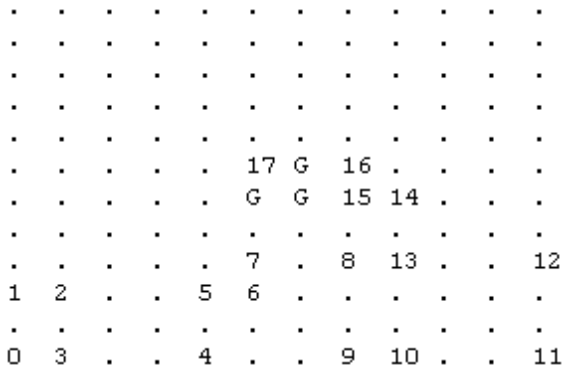

Fig. 7: Optimal path follow by the robot in maze 1

In the second maze the robot follow this locations as the optimal path:
(0, 0), (0, 3), (2, 3), (2, 2), (1, 2), (1, 0), (4, 0), (5, 0), (5, 3), (8, 3), (8, 1), (10, 1), (10, 2), (12, 2), (12, 5), (13, 5), (13, 6), (11, 6), (11, 9), (8, 9), (8, 8), (6, 8), (6, 7)

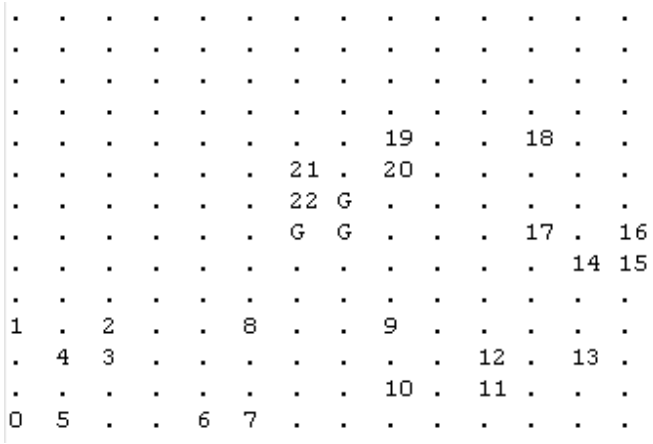

Fig. 8: Optimal path follow by the robot in maze 2

In the third maze the robot follow this locations as the optimal path:
(0, 0), (0, 3), (2, 3), (2, 4), (0, 4), (0, 7), (0, 9), (0, 12), (0, 15), (3, 15), (6, 15), (7, 15), (7, 12), (8, 12), (8, 13), (9, 13), (12, 13), (12, 10), (14, 10), (14, 9), (11, 9), (11, 7), (10, 7), (10, 6), (8, 6), (8, 8)

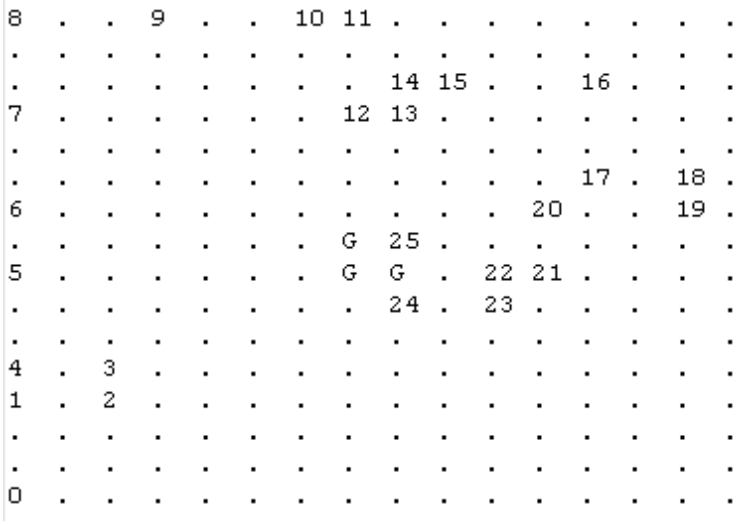

Fig. 9: Optimal path follow by the robot in maze 3

These are the results for the first try to solve the problem using random moves in maze 1.

Trial	**Goal in run 1?**	**Moves in run 1**	**Coverage**	**Goal in run 2?**	**Moves in run 2**	**Path length in run 2**	**Score**
1	False	1000	92.36%	False	-	-	-
2	True	779	86.11%	True	17	32	43
3	False	1000	94.44%	False	-	-	-
4	True	548	79.86%	True	18	35	36.3
5	False	1000	90.28%	False	-	-	-
6	False	1000	92.36%	False	-	-	-
7	True	790	91.67%	True	18	35	44.37

8	True	91	27.78%	True	17	32	20.07
9	True	858	94.44%	True	17	33	45.63
10	True	863	91.67%	True	18	35	46.8
Total	**60%**	**792.9**	**84.1%**	**60%**	**17.5**	**33.67**	**39.36**

These are the results for the first try to solve the problem using random moves in maze 2.

Trial	**Goal in run 1?**	**Moves in run 1**	**Coverage**	**Goal in run 2?**	**Moves in run 2**	**Path length in run 2**	**Score**
1	False	1000	85.20%	False	-	-	-
2	False	1000	85.20%	False	-	-	-
3	False	1000	54.59%	False	-	-	-
4	False	1000	57.14%	False	-	-	-
5	False	1000	53.06%	False	-	-	-
6	True	749	75.51%	True	22	44	47
7	True	859	71.94%	True	26	45	54.67
8	True	273	46.45%	True	28	50	37.13
9	True	911	73.47%	True	22	44	52.4
10	False	1000	66.33%	False	-	-	-
Total	**40%**	**879.2**	**66.89%**	**40%**	**24.5**	**45.75**	**47.8**

These are the results for the first try to solve the problem using random moves in maze 3.

Trial	**Goal in run 1?**	**Moves in run 1**	**Coverage**	**Goal in run 2?**	**Moves in run 2**	**Path length in run 2**	**Score**
1	True	114	20.31%	True	30	54	33.83
2	True	365	49.22%	True	25	52	37.2
3	False	1000	53.51%	False	-	-	-
4	True	499	53.51%	True	29	55	45.67
5	True	356	43.36%	True	27	56	38.9
6	False	1000	66.8%	False	-	-	-
7	True	855	71.09%	True	31	63	59.53
8	False	1000	48.44%	False	-	-	-
9	True	786	62.11%	True	28	63	54.23
10	True	640	52.34%	True	27	55	48.37
Total	**70%**	**661.5**	**52.07**	**70%**	**28.14**	**56.86**	**45.39**

These are the results using dead end path detection in maze 1.

Trial	Goal in run 1?	Moves in run 1	Coverage	Goal in run 2?	Moves in run 2	Path length in run 2	Score
1	False	1000	93.05%	False	-	-	-
2	False	1000	94.44%	False	-	-	-
3	True	955	86.81%	True	17	32	48.87
4	False	1000	91.67%	False	-	-	-
5	True	770	93.05%	True	17	32	42.7
6	True	970	88.19%	True	19	32	51.37
7	False	1000	70.14%	False	-	-	-
8	False	1000	93.75%	False	-	-	-
9	False	1000	92.36%	False	-	-	-
10	True	668	86.11%	True	17	33	39.3
Total	**40%**	**936.3**	**88.96%**	**40%**	**17.5**	**32.25**	**45.56**

These are the results using dead end path detection in maze 2.

Trial	Goal in run 1?	Moves in run 1	Coverage	Goal in run 2?	Moves in run 2	Path length in run 2	Score
1	True	824	61.22%	True	26	48	53.5
2	True	396	58.67%	True	22	43	35.23
3	False	1000	74.49%	False	-	-	-
4	False	1000	72.96%	False	-	-	-
5	False	1000	76.02%	False	-	-	-
6	False	1000	63.77%	False	-	-	-
7	True	457	62.24%	True	31	53	46.27
8	True	794	69.39%	True	29	48	55.5
9	False	1000	80.1%	False	-	-	-
10	True	455	53.57%	True	28	51	43.2
Total	**50%**	**792.6**	**67.24%**	**50%**	**27.2**	**48.6**	**46.74**

These are the results using dead end path detection in maze 3.

Trial	Goal in run 1?	Moves in run 1	Coverage	Goal in run 2?	Moves in run 2	Path length in run 2	Score
1	True	490	61.33%	True	34	70	50.36
2	True	247	39.84%	True	28	52	36.27
3	False	1000	72.27%	False	-	-	-
4	True	582	60.16%	True	26	53	45.43
5	True	484	65.23%	True	27	56	43.17

6	False	1000	80.86%	False	-	-	-
7	False	1000	42.19%	False	-	-	-
8	True	900	79.3%	True	29	60	59.03
9	False	1000	79.69%	False	-	-	-
10	False	1000	67.19%	False	-	-	-
Total	**50%**	**770.3**	**64.81%**	**50%**	**28.8**	**58.2**	**46.85**

These are the results using counting of locations visited in maze 1.

Trial	**Goal in run 1?**	**Moves in run 1**	**Coverage**	**Goal in run 2?**	**Moves in run 2**	**Path length in run 2**	**Score**
1	True	436	97.22%	True	17	33	31.57
2	True	427	97.22%	True	17	33	31.27
3	True	162	79.86%	True	18	34	23.43
4	True	530	95.83%	True	17	32	34.7
5	True	358	93.05%	True	17	33	28.97
6	True	95	52.08%	True	27	51	30.2
7	True	240	86.11%	True	17	32	25.03
8	True	168	81.25%	True	17	33	22.63
9	True	319	95.14%	True	17	33	27.67
10	True	322	97.22%	True	17	33	27.77
Total	**100%**	**305.7**	**87.5%**	**100%**	**18.1**	**34.7**	**28.32**

These are the results using counting of locations visited in maze 2.

Trial	**Goal in run 1?**	**Moves in run 1**	**Coverage**	**Goal in run 2?**	**Moves in run 2**	**Path length in run 2**	**Score**
1	True	136	54.59%	True	27	48	31.57
2	True	102	40.82%	True	28	50	31.43
3	True	194	60.20%	True	24	45	30.5
4	True	143	53.06%	True	23	43	27.8
5	True	134	47.96%	True	22	44	26.5
6	True	372	87.75%	True	23	44	35.43
7	True	134	51.02%	True	27	46	31.5
8	True	173	61.22%	True	22	44	27.8
9	True	217	73.47%	True	22	43	29.27
10	True	173	59.69%	True	29	50	34.8
Total	**100%**	**177.8**	**58.98%**	**100%**	**24.7**	**45.7**	**30.66**

These are the results using counting of locations visited in maze 3.

Trial	Goal in run 1?	Moves in run 1	Coverage	Goal in run 2?	Moves in run 2	Path length in run 2	Score
1	True	295	67.19%	True	25	52	34.87
2	True	246	68.36%	True	29	55	37.23
3	True	302	76.95%	True	25	52	35.1
4	True	50	18.75%	True	33	63	34.7
5	True	139	47.26%	True	29	56	33.67
6	True	175	58.59%	True	29	55	34.87
7	True	124	39.84%	True	37	70	41.17
8	True	94	30.08%	True	39	72	42.17
9	True	156	46.48%	True	29	52	34.23
10	True	142	37.89%	True	29	55	33.77
Total	**100%**	**172.3**	**49.14%**	**100%**	**30.4**	**58.2**	**36.18**

These are the results using heuristic values in maze 1.

Trial	Goal in run 1?	Moves in run 1	Coverage	Goal in run 2?	Moves in run 2	Path length in run 2	Score
1	True	466	88.89%	True	17	30	32.57
2	False	1000	94.44%	False	-	-	-
3	True	835	94.44%	True	17	32	44.87
4	True	507	84.03%	True	19	32	35.93
5	True	544	94.44%	True	17	33	35.17
6	True	545	88.89%	True	17	33	35.2
7	True	469	84.72%	True	17	33	32.67
8	False	1000	94.44%	False	-	-	-
9	False	1000	86.11%	False	-	-	-
10	False	1000	94.44%	False	-	-	-
Total	**60%**	**736.6**	**90.48%**	**60%**	**17.33**	**32.17**	**36.07**

These are the results using heuristic values in maze 2.

Trial	Goal in run 1?	Moves in run 1	Coverage	Goal in run 2?	Moves in run 2	Path length in run 2	Score
1	False	1000	88.77%	False	-	-	-
2	False	1000	87.75%	False	-	-	-
3	True	322	55.61%	True	22	43	32.77
4	True	748	88.77%	True	22	44	46.97

5	False	1000	58.67%	False	-	-	-
6	False	1000	88.77%	False	-	-	-
7	True	796	75%	True	22	43	48.57
8	True	401	66.33%	True	23	43	36.4
9	True	274	51.02%	True	24	46	33.17
10	True	717	83.67%	True	24	45	47.93
Total	**60%**	**725.8**	**74.43%**	**60%**	**22.83**	**44**	**40.97**

These are the results using heuristic values in maze 3.

Trial	**Goal in run 1?**	**Moves in run 1**	**Coverage**	**Goal in run 2?**	**Moves in run 2**	**Path length in run 2**	**Score**
1	True	457	73.05%	True	27	55	42.26
2	True	213	56.25%	True	27	56	34.13
3	True	405	66.41%	True	28	53	41.53
4	False	1000	82.42%	False	-	-	-
5	True	822	89.84%	True	25	51	52.43
6	True	475	80.08%	True	25	51	40.87
7	True	778	85.55%	True	25	52	50.97
8	True	576	80.86%	True	26	55	45.23
9	True	995	80.86%	False	-	-	-
10	True	292	64.06%	True	25	52	34.77
Total	**90%**	**601.3**	**75.94%**	**80%**	**26**	**53.12**	**42.77**

It doesn't observe difference significantly that an algorithm works better in a specific maze than others.

Justification

In the run 1 the best performance in score is achieved counting the locations visited in all mazes because the number of moves required to reach the goal is lower than other. However, the coverage is low hence it's hard to get the optimal path in run 2 for that reason requires more moves in run 2. This is a table summaring the average results.

Maze	**Random move. Moves in run 1 / run 2**	**Random move. Score**	**Dead end path detection. Moves in run 1 / run 2**	**Dead end path detection. Score**	**Count visits. Moves in run 1 / run 2**	**Count visits. Score**	**Heuristic values. Moves in run 1 / run 2**	**Heuristic values. Scores**	**A*. Moves in run 2**
1	792.9 / 17.5	39.36	936.3 / 17.5	45.56	**305.7** / 18.1	**28.32**	736.6 / 17.33	36.07	**17**

2	879.2 / 24.5	47.8	792.6 / 27.2	46.74	**177.8** / 24.7	**30.66**	725.8 / 22.83	40.97	**22**
3	661.5 / 28.14	45.39	770.3 / 28.8	46.85	**172.3** / 30.4	**36.18**	601.3 / 26	42.77	**25**

In the tests the unique technique that always accomplished the benchmark about to reach the goal in less than 1000 moves in all mazes was the counting of locations visited. The others fails sometimes. This is a table showing percentage of successful.

Maze	**Random move. Goal reached in less than 1000 moves?**	**Dead end path detection. Goal reached in less than 1000 moves?**	**Count visits. Goal reached in less than 1000 moves?**	**Heuristic values. Goal reached in less than 1000 moves?**
1	60%	40%	100%	60%
2	40%	50%	100%	60%
3	70%	50%	100%	80%

Conclusion

Free-Form Visualization

The less number of moves in maze 1 was gotten in a trial of counting visits in run 1 but the score isn't so good (30.2) because in run 2 a* search follows a long path because in run 1 only explores almost the half of maze. This is was happened in run 2.

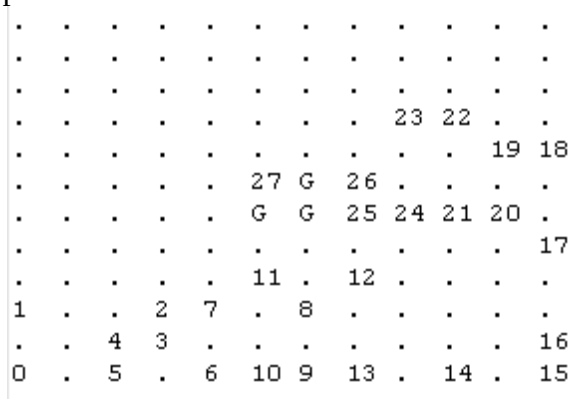

Fig. 10: Final result in run 2 for a trial with 52.08% of coverage in run 1.

This show the importance in exploration in first run and few moves in run 1 it doesn't ensure a low score. The score measures very well a fast exploration but finding a good path.

Reflection

This problem about find the best path to reach a goal from a initial point was solved dividing it in two parts: first, exploring the place (in this case a maze) testing different techniques (dead end path detection, counting visits and adding heuristic values) and then using that information to apply an algorithm to reach the goal in the best way according the explored. I think this was a good way to face this kind of problem in path planning.
A difficulty for me in this project was to understand the starter code, for example, how the score is calculated and why in that way, how the robot moves for the maze. Also, it was hard to fit the a* implementation found with the starter code because for example, in the a* algorithm the walls were thought as nodes in a grid no like edges.
An interesting code was the way to create mazes codified in only one number and this is enough to know where there are walls or it's open to move the robot.

Improvement

The true challenge is to create a simulation more seemed to the real micromouse challenge, performing moves in real time and using continuos values for everything (sensors, displacement and time). This would be a cool improvement and would impact the solution. The robot could move diagonally and sensors could give two new measure in left-up and right-up of heading and rotation could be done in 45 degrees. The score should change to time in seconds because number of moves isn't so precise in real time.
I think the techniques used here in run 1 (dead end path detection, counting visits, etc) it could work in real time with few adjusts and maybe combine them it can generate good results in exploration in the first run.

It would be interesting to test other algorithms used in motion planning instead of a* like d*, rapidly-exploring random tree or probabilistic roadmap to see if they give better results.

References

[1] Red Blob Games. http://www.redblobgames.com/pathfinding/a-star/implementation.html