Jules Morin

MazeSolver Report

I - Environment of the Artificial Intelligence

The main objective of this project is to create an artificial intelligence able to learn how to solve a given maze. To achieve this project, I based most of my choices on the A-Mazer with Genetic Algorithm paper published by Nitin S. Choubey in November 2012.

I-I The Maze

The first objective was then to create this "maze environment" where the AI will evolve.
I decided to make this whole project with *Python* language and the *Pygame* library to create a graphical interface.
I imagined my Maze as a grid of multiple cells. Each of this cell is a square with the possibility to have a "wall" on his top, right, down or left. The list of possible cells is shown bellow in **Figure 1**.

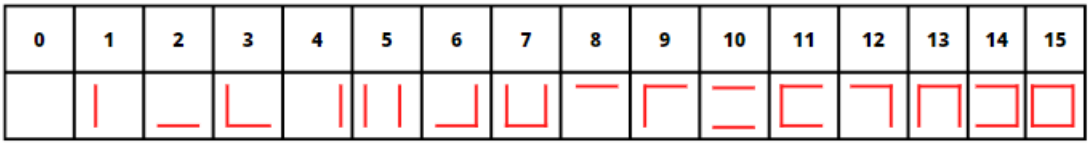
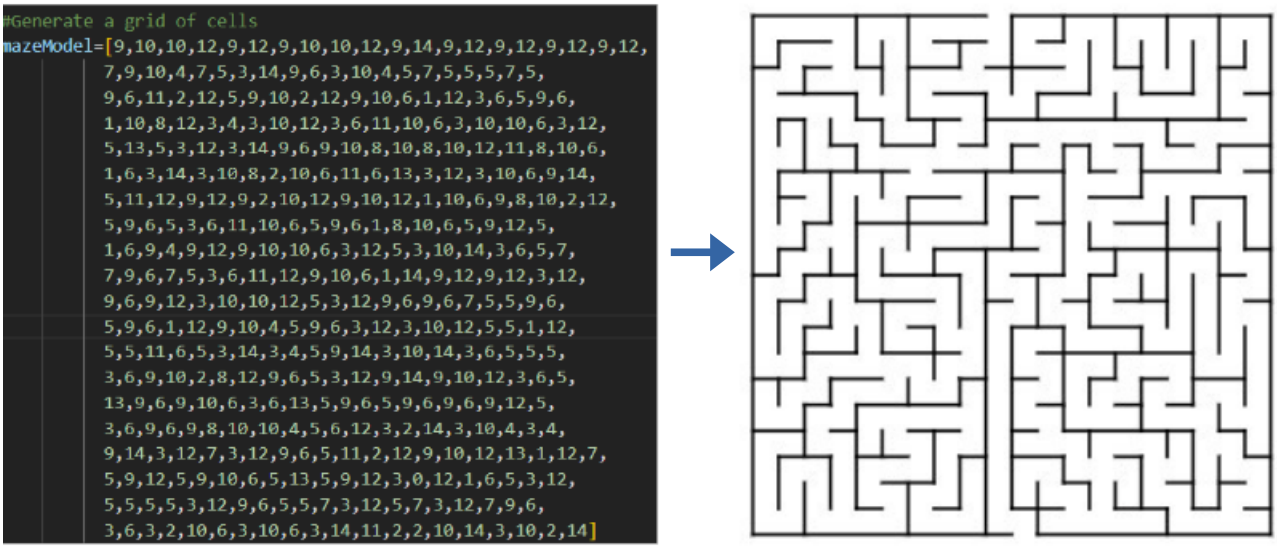


Figure 1: All possible cells with there attributed index

With this Cell object done, I implemented the generation of a maze can from a list of Cells index as you can see in **Figure 2**.



```
#Generate a grid of cells
mazeModel=[9,10,10,12,9,12,9,10,10,12,9,14,9,12,9,12,9,12,9,12,
           7,9,10,4,7,5,3,14,9,6,3,10,4,5,7,5,5,5,7,5,
           9,6,11,2,12,5,9,10,2,12,9,10,6,1,12,3,6,5,9,6,
           1,10,8,12,3,4,3,10,12,3,6,11,10,6,3,10,10,6,3,12,
           5,13,5,3,12,3,14,9,6,9,10,8,10,8,10,12,11,8,10,6,
           1,6,3,14,3,10,8,2,10,6,11,6,13,3,12,3,10,6,9,14,
           5,11,12,9,12,9,2,10,12,9,10,12,1,10,6,9,8,10,2,12,
           5,9,6,5,3,6,11,10,6,5,9,6,1,8,10,6,5,9,12,5,
           1,6,9,4,9,12,9,10,10,6,3,12,5,3,10,14,3,6,5,7,
           7,9,6,7,5,3,6,11,12,9,10,6,1,14,9,12,9,12,3,12,
           9,6,9,12,3,10,10,12,5,3,12,9,6,9,6,7,5,5,9,6,
           5,9,6,1,12,9,10,4,5,9,6,3,12,3,10,12,5,5,1,12,
           5,5,11,6,5,3,14,3,4,5,9,14,3,10,14,3,6,5,5,5,
           3,6,9,10,2,8,12,9,6,5,3,12,9,14,9,10,12,3,6,5,
           13,9,6,9,10,6,3,6,13,5,9,6,5,9,6,9,6,9,12,5,
           3,6,9,6,9,8,10,10,4,5,6,12,3,2,14,3,10,4,3,4,
           9,14,3,12,7,3,12,9,6,5,11,2,12,9,10,12,13,1,12,7,
           5,9,12,5,9,10,6,5,13,5,9,12,3,0,12,1,6,5,3,12,
           5,5,5,5,3,12,9,6,5,5,7,3,12,5,7,3,12,7,9,6,
           3,6,3,2,10,6,3,10,6,3,14,11,2,2,10,14,3,10,2,14]
```

Figure 2: Maze model from a list of cell index

I-II The Individuals evolving in the maze

To see how the AI evolve in the created maze, I need to create an Individual object, able to move by following the rules of this environment (wall collision, 4 way to move).
In preparation of the genetic algorithm described bellow, this individual has to specific attributes such as the historic of all his previous moves.

II - Genetic algorithm and methodology

II-1 Introduction

My genetic algorithm will produce multiple generations of individuals. Between each generation, all the individual will calculate their Fitness, the score from 0 to 20000 to determine how close they are to go out from the Maze. All the best individuals will then be selected to participate to the next generation.
The number of move of each individual will increase for each generation. For example, if an individual made 20 random moves in Generation 1, if he is selected, in the 2nd Generation he will then do the same 20 previous moves (with little random changes because of the mutation) and 20 new random moves (Total =40).
The historic of moves previously quoted will be the chromosome structure of my individuals, the memory of what he made during the previous generation.

II-2 Fitness function

The fitness function I chose to get the fitness value of my individuals at the end of a generation is the following the same as in the based on paper from Nitin S. Choubey:

$$FV = \frac{CC - SC}{FC - CC} * 100$$

With FV=Fitness Value, CC = Current Cell, SC = Starting Cell, FC = Final Cell

II-2.1 Calculate the current cell score

The first problem I encountered with this fitness function is what value I had to put in CC.
How can I give an index to every cell of my maze to be sure that being close to the end will give you a better score than being close the starting cell?

You can see the first model I choose in the **Figure 3** bellow.
In this model, the starting cell is in 0 and the Final cell is the 99.
The problem with this model is that with the fitness calcul I make, being in indx 9 is worst than being in 90 even if in fact you are in both case at the same direct distance of the start.

I then decided to use like a x and y coordinate system to solve this as you can see in **Figure 4**.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

Figure 3: First Cell value model

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

Figure 4: Second Cell value model

The final model I decided to use was convenient to create from the old model I had and was for me the most easy to understand. It I represented in the **Figure 5** with a 20x20 size Maze. Each cell value correspond to the combination of his X and Y coordinate.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190
1	10	11	21	31	41	51	61	71	81	91	101	111	121	131	141	151	161	171	181	191
2	20	21	22	32	42	52	62	72	82	92	102	112	122	132	142	152	162	172	182	192
3	30	31	32	33	43	53	63	73	83	93	103	113	123	133	143	153	163	173	183	193
4	40	41	42	43	44	54	64	74	84	94	104	114	124	134	144	154	164	174	184	194
5	50	51	52	53	54	55	65	75	85	95	105	115	125	135	145	155	165	175	185	195
6	60	61	62	63	64	65	66	76	86	96	106	116	126	136	146	156	166	176	186	196
7	70	71	72	73	74	75	76	77	87	97	107	117	127	137	147	157	167	177	187	197
8	80	81	82	83	84	85	86	87	88	98	108	118	128	138	148	158	168	178	188	198
9	90	91	92	93	94	95	96	97	98	99	109	119	129	139	149	159	169	179	189	199
10	100	101	102	103	104	105	106	107	108	109	1010	1110	1210	1310	1410	1510	1610	1710	1810	1910
11	110	111	112	113	114	115	116	117	118	119	1110	1111	1211	1311	1411	1511	1611	1711	1811	1911
12	120	121	122	123	124	125	126	127	128	129	1210	1211	1212	1312	1412	1512	1612	1712	1812	1912
13	130	131	132	133	134	135	136	137	138	139	1310	1311	1312	1313	1413	1513	1613	1713	1813	1913
14	140	141	142	143	144	145	146	147	148	149	1410	1411	1412	1413	1414	1514	1614	1714	1814	1914
15	150	151	152	153	154	155	156	157	158	159	1510	1511	1512	1513	1514	1515	1615	1715	1815	1915
16	160	161	162	163	164	165	166	167	168	169	1610	1611	1612	1613	1614	1615	1616	1716	1816	1916
17	170	171	172	173	174	175	176	177	178	179	1710	1711	1712	1713	1714	1715	1716	1717	1817	1917
18	180	181	182	183	184	185	186	187	188	189	1810	1811	1812	1813	1814	1815	1816	1817	1818	1918
19	190	191	192	193	194	195	196	197	198	199	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919

Figure 5: Final Cell value model

To see if this Fitness value formula was correct I decided to make a simulation to get a graphical representation of the fitness value depending on you position in the maze. You can see this representation in the **Figure 6** for the First Cell value model and **Figure 7** for the final one.

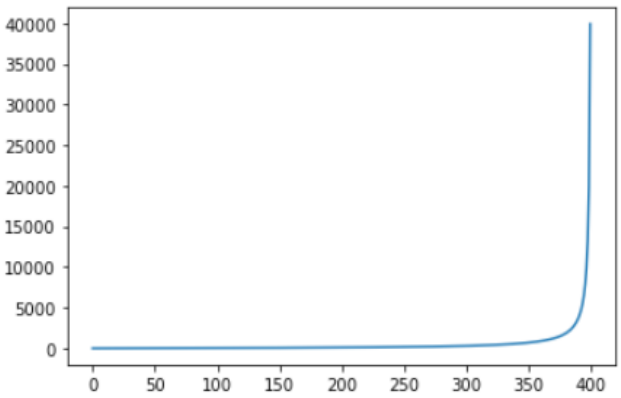


Figure 6: Fitness Value Y from cell position X with first cell value model

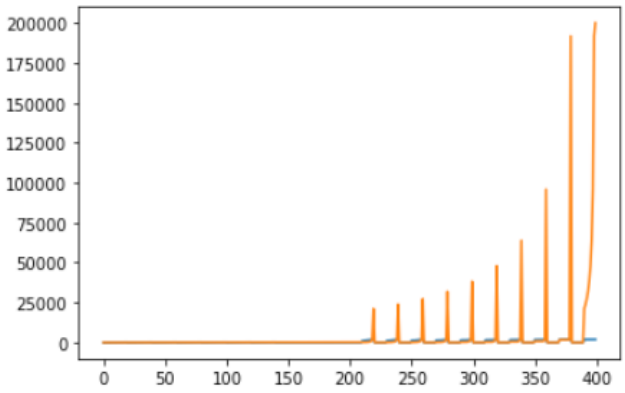


Figure 7: Fitness Value Y from cell position X with second cell value model

II-2.2 Invalid moves and collision

We can see that the invalids moves like collision to a wall are not parameter of the used fitness function of the Nitin S. Choubey paper.
Three solution can be imagined to improve the current system:

- Because collision to the walls are implemented so the individuals can't cross a wall, every movement in the history of an individual that lead to a collision could be memorized. After that, we would have to determinate a way to integrate this number of collision directly into the fitness value calculation so people who do less collisions would be considered better than people who don't.
- Change the move function of an individual for it to check the current cell's walls and allow him to move in directions empty of wall.

Integration of invalid moves into this system would lead to a smaller number of moves needed to solve the maze because useless ones won't be counted.

II-3 Crossover Operation

For the crossover operation, at the end of the generation, individuals are sorted in descending order of their fitness value. Half of the population is erased and "children" are created from couples of parents. For example, in my individuals list, the first and the second parents will form a couple and generate 2 children. The chromosome structure of a child will be filled by the combination of his couple parents.
The first child chromosomes will result from the addition of 2 parents chromosome modulo 4.
The second child chromosomes will result from the substraction of 2 parents chromosome modulo 4.

In my project, the chromosome is represented by 0,1,2 or 3 where each of these number represent a move direction:
0 → up
1 → right
2 → down
3 → left

This crossover operation is represented in **Figure 8**.

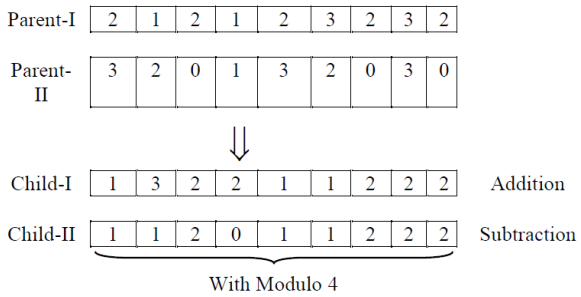


Figure 8: Crossover Operation

II-4 Mutations

For the mutation operation, I decided to use a different system than the one presented by Nitin S. Choubey. Instead of selecting a random chromosome and change it randomly I made a system where each chromosome has a fixed by the user percentage of chance to chose randomly.

III – Results

III-1 First results

To consider that result where good enough I fixed my first objective to the same given in Nitin S. Choubey Paper:
Being able to find parameters for an individual to complete the Maze in less than the dimension of the maze (20x20 = 400).
The tested mazes only have one unique path to the solution.
With the Maze I created, I managed to do it with a population of 96 individuals and a mutation rate of 0 to 6% in a bit less than the maximum number of moves.
The problem I can see is that my population is sprayed all around the maze where I was more expecting a concentration of it on the good path to go to the end.
I think that this come from the fact that I keep to much parents (half of the initial population). I think that only keeping the 10 best % of my population and create children from them with more mutation rate would solve this problem.
Figure 9 shows this simulation just before the maze get solved.

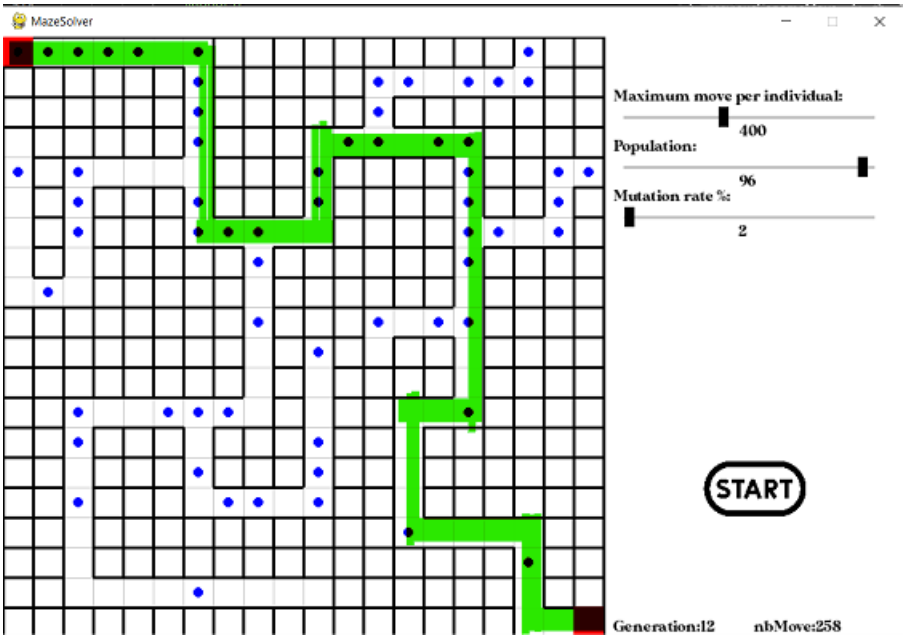


Figure 9: Final result

III-2.1 Corrections

I decided to implement the proposed correction in III-1 about modifying the number of parents you keep.
I added a slider “parents kept (%)”
To check if it was working as before with the 50 % population kept, I first tried to run with 50% to see if I obtain the same results. This step was confirmed so I tried with other parameters and results changed hugely:
Some simulations don’t work with specific sliders parameters but I managed to make this one run: With a 20% parents kept, a mutation rate included between 0 to 10% and an original population of 100, the solution of the maze is found in 4 generations on average (compared to the average 18 generations with the 50% simulation).

III-2.1 Comparison with other heuristic method

Being able to compare the obtained results to other existing method is mandatory to see if our solution is good.
I decided to compare to one of the simplest heuristic method: always following the wall on your right.
To try this solution, you just have to set the Population number to 2 and click on Heuristic test button.

Here is a table of the results of AI compared to this heuristic method with the III-1 part maze model:

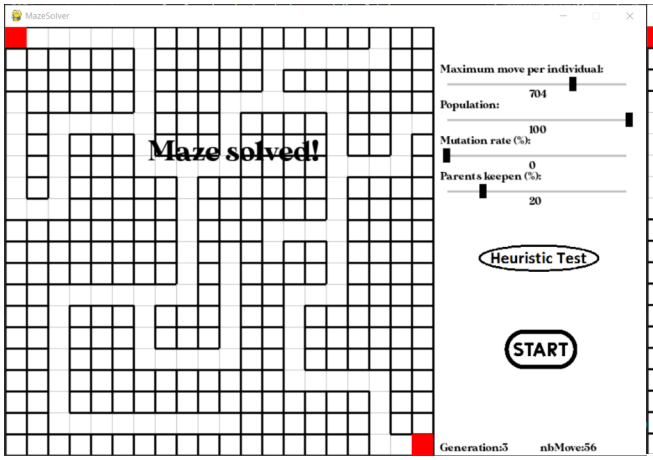


Figure 10: Final result with 20% parents

Number total of moves:
20+40+56=116

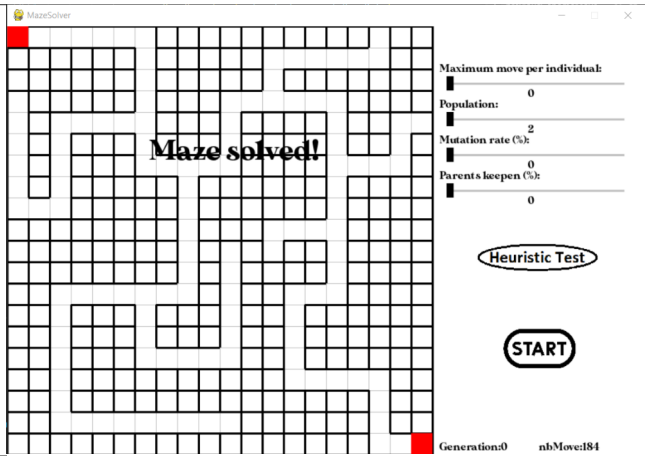


Figure 11: Final result follow right wall

Number total of moves:
184

With this first result, we can see that genetic algorithm look slightly better.
To imagine how this situation would evolve, I tried both solution with a way harder maze:

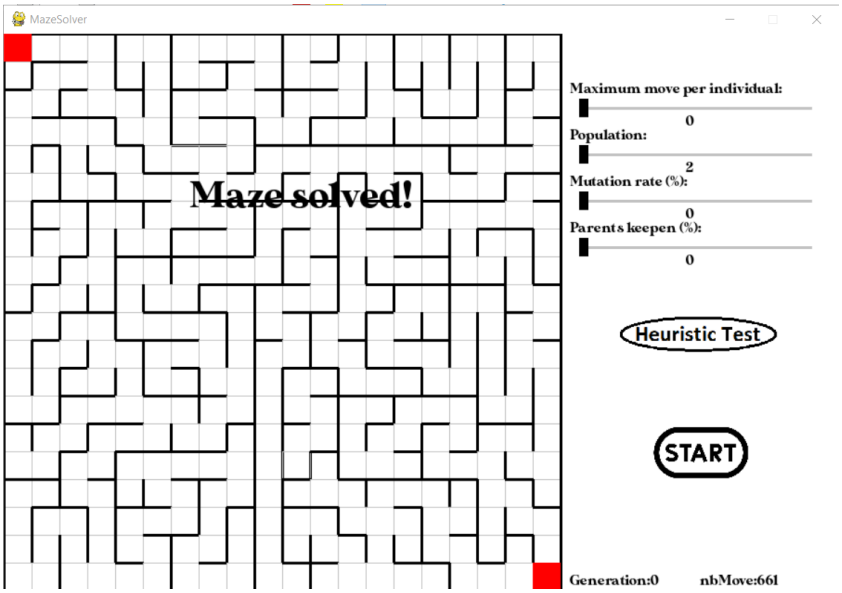


Figure 12: Following right wall with harder maze

Unfortunately, I didn't managed to make my genetic algorithm finish these model but here is what I suppose:
We can see that the number total of move changed a lot (from 184 moves to 661) just by changing the complexity of the maze (without even changing his dimensions)

I would say that this heuristic method total move number would evolve exponentially if the maze dimension/complexity increase while the genetic algorithm solution result would evolve slower.

Presentation link:
https://www.canva.com/design/DAFZVyHvJwM/_cYaHlk_eloPxdplMByY6Q/view?utm_content=DAFZVyHvJwM&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton
GitHub of the project: https://github.com/j8morin/mazesolver.git