Analysis

March 24, 2023

1 Consideraciones de diseño

Se especificó manualmente la semilla de aleatoriedad de Python para obtener resultados comparables en los escenarios expuestos y para poder reproducir los resultados de los análisis. Adicionalmente, se usaron hyper parametros alpha = 0.3, épsilon = 0.1 y gamma = 0.6 como base de todos los escenarios, con modificaciones para el escenario de taxi y laberinto. Finalmente se adiciono una restricción máxima de 200 pasos para acelerar la convergencia de los escenarios.

[ ]:
```
%pip install matplotlib pandas --q
```

```
Note: you may need to restart the kernel to use updated packages.
```

[ ]:
```
from board import Coordinates, generate_grid_world_board, Grid,␣
 ↪generate_labyrinth, generate_taxi, TaxiGrid, GridBlockedPaths
from math import ceil
from matplotlib.pyplot import Rectangle, subplots, rcParams
from q_learning import QLearning, QLearningTaxi
from pandas import DataFrame, set_option

set_option('display.max_rows', 500)
set_option('display.max_columns', 500)
set_option('display.width', 150)

rcParams['figure.figsize'] = [12, 8]
rcParams['figure.dpi'] = 100


def plot_scenario(scenario: QLearning):
    _, axes = subplots()
    axes.set_aspect('equal')
    axes.set_xlim(0, scenario.environment.dimensions.x)
    axes.set_ylim(scenario.environment.dimensions.y, 0)
    for x in range(scenario.environment.dimensions.x):
        for y in range(scenario.environment.dimensions.y):
            coordinates = Coordinates(x, y)
            if coordinates in scenario.environment.board:
                value = ceil((scenario.get_value(coordinates)) * 100) / 100
                color = 'white'
```

```
                qvalue = scenario.get_policy_qvalue(coordinates)
                if isinstance(scenario, QLearningTaxi):
                    if coordinates == scenario.environment.objective:
                        color = "green"
                    elif coordinates == scenario.environment.passenger:
                        color = "blue"
                    elif coordinates in scenario.environment.objectives:
                        color = "orange"
                else:
                    if coordinates == scenario.environment.objective:
                        color = "green"
                    elif coordinates == scenario.environment.start:
                        color = "blue"
                    elif value < 0:
                        color = "red"
                if qvalue != None:
                    qvalue = ceil(qvalue * 100) / 100
                    action = scenario.get_policy(coordinates)
                    action = "OUT" if action == None else action.name
                else:
                    qvalue = 0
                    action = "None"
                axes.add_patch(Rectangle((x, y), 1, 1, facecolor=color))
                if isinstance(scenario, QLearningTaxi):
                    text = "Con:{}\nQ:{:.2f}\nSin:{}\nQ:{:.2f}".format(scenario.
 ↪get_policy_passenger(
                        coordinates).name, scenario.
 ↪get_policy_qvalue_passenger(coordinates), action, qvalue)
                else:
                    text = "{}\nQ:{:.2f}\n{:.2f}".format(action, qvalue, value)
                axes.text(x + 0.5, y + 0.5, text, ha='center', va='center')
            else:
                axes.add_patch(Rectangle((x, y), 1, 1, facecolor='gray'))
    for (first, second) in scenario.environment.blocked_paths:
        if first.x == second.x:
            x = [first.x, first.x + 1]
            y = [second.y, second.y] if first.y < second.y else [first.y, first.
 ↪y]
            axes.plot(x, y, color="black")
        elif first.y == second.y:
            y = [first.y, second.y + 1]
            x = [second.x, second.x] if first.x < second.x else [first.x, first.
 ↪x]
            axes.plot(x, y, color="black")


def generate_q_table(scenario: QLearning) -> DataFrame:
```

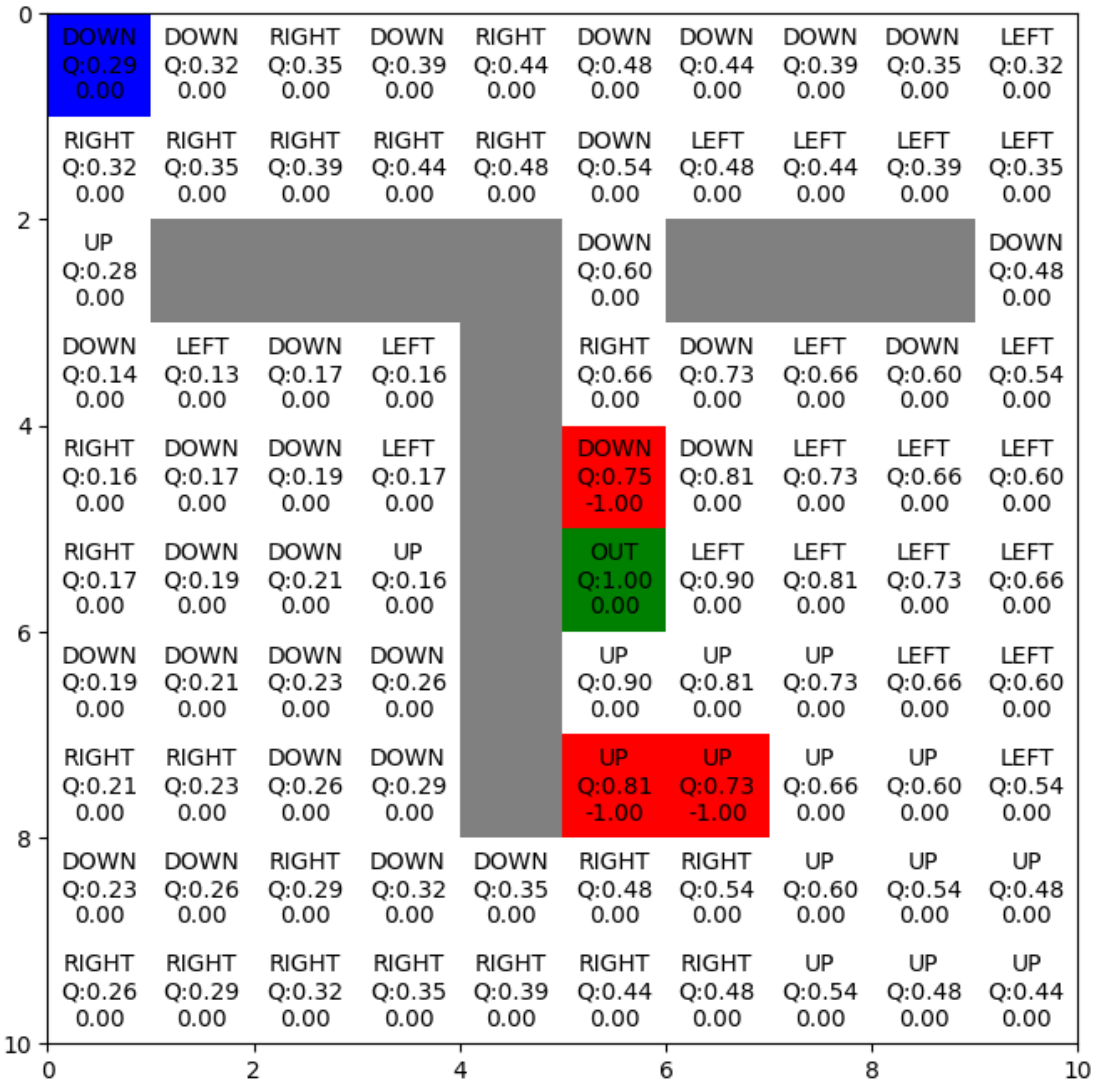
```
    q_table = {"Estado": [], }
    for action in scenario.environment.get_actions():
        q_table[action.name] = []
    q_table["Objetivo"] = []
    for state in sorted(scenario.Q, key=lambda x: (x.x, x.y)):
        if len(scenario.Q[state]) < 2:
            continue
        q_table["Objetivo"].append(
            "Sí" if state == scenario.environment.objective else "No")
        q_table["Estado"].append(f"({state.x}, {state.y})")
        for action in scenario.Q[state]:
            q_table[action.name].append(scenario.Q[state][action])
    return DataFrame(q_table)


random_seed = 5657656
```

1.1 Q-learning

1.1.1 Gridworld

[ ]:
```
grid_world_board, grid_world_objective, grid_world_start =␣
 ↪generate_grid_world_board()
grid_world = Grid(grid_world_board, grid_world_objective, grid_world_start)
q_learning = QLearning(grid_world, random_seed, gamma=0.9)
print(
    f"El agente toma {q_learning.run()} muestras para converger valores a 3␣
 ↪decimales")
plot_scenario(q_learning)
```

```
El agente toma 22444 muestras para converger valores a 3 decimales
```



[ ]:
```
generate_q_table(q_learning)
```

[ ]:

	Estado	DOWN	LEFT	RIGHT	UP	OUT	Objetivo
0	(0, 0)	0.282430	0.067349	0.070108	0.026589	0.072108	No
1	(0, 1)	0.135244	0.233151	0.313811	0.122934	0.142074	No
2	(0, 2)	0.121577	0.101864	0.108350	0.274758	0.107061	No
3	(0, 3)	0.135085	0.120629	0.109053	0.109300	0.120221	No
4	(0, 4)	0.150036	0.135034	0.150095	0.121326	0.135055	No
5	(0, 5)	0.153037	0.108522	0.166772	0.097871	0.145073	No
6	(0, 6)	0.185302	0.153034	0.137188	0.130269	0.085054	No
7	(0, 7)	0.178379	0.182639	0.205891	0.166128	0.160808	No
8	(0, 8)	0.228768	0.105004	0.190288	0.140811	0.061767	No
9	(0, 9)	0.219721	0.173841	0.254187	0.156456	0.201854	No

10	(1, 0)	0.313811	0.130272	0.238465	0.173166	0.073633	No
11	(1, 1)	0.306096	0.219334	0.348678	0.193682	0.294247	No
12	(1, 3)	0.098612	0.121577	0.062004	0.070778	0.107255	No
13	(1, 4)	0.166772	0.135085	0.155902	0.109419	0.150095	No
14	(1, 5)	0.185302	0.150095	0.181431	0.150095	0.166772	No
15	(1, 6)	0.205891	0.166772	0.205674	0.166772	0.185302	No
16	(1, 7)	0.228768	0.185302	0.228768	0.185302	0.205891	No
17	(1, 8)	0.254187	0.156457	0.208608	0.202565	0.225832	No
18	(1, 9)	0.253925	0.228049	0.282430	0.221737	0.253572	No
19	(2, 0)	0.278487	0.187127	0.348678	0.287967	0.287962	No
20	(2, 1)	0.229082	0.264984	0.387420	0.313811	0.321258	No
21	(2, 3)	0.166772	0.106263	0.100274	0.118879	0.132683	No
22	(2, 4)	0.185302	0.150095	0.145123	0.145292	0.159907	No
23	(2, 5)	0.205891	0.166772	0.125843	0.162602	0.158421	No
24	(2, 6)	0.228768	0.182061	0.222875	0.173226	0.202202	No
25	(2, 7)	0.254187	0.205891	0.254182	0.205891	0.228768	No
26	(2, 8)	0.282430	0.228768	0.282430	0.228768	0.254187	No
27	(2, 9)	0.282411	0.254178	0.313811	0.253541	0.280734	No
28	(3, 0)	0.387420	0.293847	0.386786	0.341073	0.346314	No
29	(3, 1)	0.386651	0.345047	0.430467	0.347867	0.387318	No
30	(3, 3)	0.062004	0.150095	0.098673	0.035673	0.063461	No
31	(3, 4)	0.103812	0.166772	0.131041	0.067818	0.135853	No
32	(3, 5)	0.099177	0.114669	0.113965	0.150094	0.077637	No
33	(3, 6)	0.254187	0.081913	0.123119	0.117432	0.186446	No
34	(3, 7)	0.282430	0.228768	0.230208	0.206147	0.248543	No
35	(3, 8)	0.313811	0.254187	0.313811	0.254186	0.282430	No
36	(3, 9)	0.313811	0.282430	0.348678	0.282430	0.313811	No
37	(4, 0)	0.219538	0.334608	0.430467	0.294401	0.355515	No
38	(4, 1)	0.429977	0.387078	0.478297	0.387396	0.430349	No
39	(4, 8)	0.348678	0.282430	0.344517	0.263894	0.295973	No
40	(4, 9)	0.348678	0.313811	0.387420	0.313811	0.348678	No
41	(5, 0)	0.478297	0.355511	0.362368	0.428424	0.219538	No
42	(5, 1)	0.531441	0.430467	0.430467	0.430467	0.478297	No
43	(5, 2)	0.590490	0.531441	0.531441	0.478297	0.531441	No
44	(5, 3)	-0.343900	0.590490	0.656100	0.531441	0.590490	No
45	(5, 4)	0.748737	-0.315625	0.729000	0.543882	0.618277	No
46	(5, 5)	0.722135	0.683910	0.615325	-0.103170	1.000000	Sí
47	(5, 6)	-0.475249	0.761491	0.539658	0.900000	0.743293	No
48	(5, 7)	0.430467	-0.586969	-0.332108	0.810000	0.227589	No
49	(5, 8)	0.254535	0.306796	0.478297	-0.453967	0.388005	No
50	(5, 9)	0.387420	0.348678	0.430467	0.430467	0.387420	No
51	(6, 0)	0.430467	0.129140	0.307655	0.255612	0.238805	No
52	(6, 1)	0.430459	0.478297	0.387418	0.387274	0.430462	No
53	(6, 3)	0.729000	0.590490	0.590490	0.656100	0.656100	No
54	(6, 4)	0.810000	-0.343900	0.656100	0.656100	0.729000	No
55	(6, 5)	0.729000	0.900000	0.729000	0.729000	0.810000	No
56	(6, 6)	-0.343940	0.809738	0.656100	0.810000	0.728036	No

57	(6, 7)	0.313424	-0.490396	0.490571	0.729000	0.514786	No
58	(6, 8)	0.430467	0.429553	0.531441	-0.337100	0.477915	No
59	(6, 9)	0.430467	0.387420	0.478297	0.478297	0.430467	No
60	(7, 0)	0.387420	0.293611	0.160026	0.177826	0.177826	No
61	(7, 1)	0.387369	0.430467	0.347023	0.348111	0.385581	No
62	(7, 3)	0.651649	0.656100	0.520933	0.566265	0.573708	No
63	(7, 4)	0.729000	0.729000	0.590490	0.590490	0.656077	No
64	(7, 5)	0.656100	0.810000	0.656100	0.656100	0.729000	No
65	(7, 6)	0.590490	0.729000	0.590490	0.729000	0.656100	No
66	(7, 7)	0.531441	-0.343900	0.531441	0.656100	0.590490	No
67	(7, 8)	0.478297	0.478297	0.478297	0.590490	0.531441	No
68	(7, 9)	0.478297	0.430467	0.430467	0.531441	0.478297	No
69	(8, 0)	0.348678	0.334608	0.279847	0.276891	0.287946	No
70	(8, 1)	0.348400	0.387420	0.268735	0.311238	0.348271	No
71	(8, 3)	0.590490	0.588528	0.476707	0.531016	0.531141	No
72	(8, 4)	0.656052	0.656100	0.531406	0.531389	0.590485	No
73	(8, 5)	0.590490	0.729000	0.584225	0.589528	0.656016	No
74	(8, 6)	0.531441	0.656100	0.531441	0.656100	0.590490	No
75	(8, 7)	0.478291	0.590490	0.478297	0.590490	0.531441	No
76	(8, 8)	0.430451	0.530149	0.430460	0.531441	0.478293	No
77	(8, 9)	0.430421	0.467604	0.387420	0.478297	0.430399	No
78	(9, 0)	0.129635	0.313811	0.266148	0.249202	0.084729	No
79	(9, 1)	0.219538	0.348583	0.076231	0.282430	0.210933	No
80	(9, 2)	0.478297	0.372824	0.219503	0.228441	0.395016	No
81	(9, 3)	0.529675	0.531441	0.464786	0.379819	0.473629	No
82	(9, 4)	0.552128	0.590490	0.500804	0.397904	0.442122	No
83	(9, 5)	0.487563	0.656100	0.503078	0.487668	0.567736	No
84	(9, 6)	0.475853	0.590490	0.520833	0.542537	0.516429	No
85	(9, 7)	0.430467	0.531441	0.478297	0.531440	0.478296	No
86	(9, 8)	0.387420	0.467612	0.430467	0.478297	0.430467	No
87	(9, 9)	0.387419	0.416710	0.387314	0.430467	0.387403	No

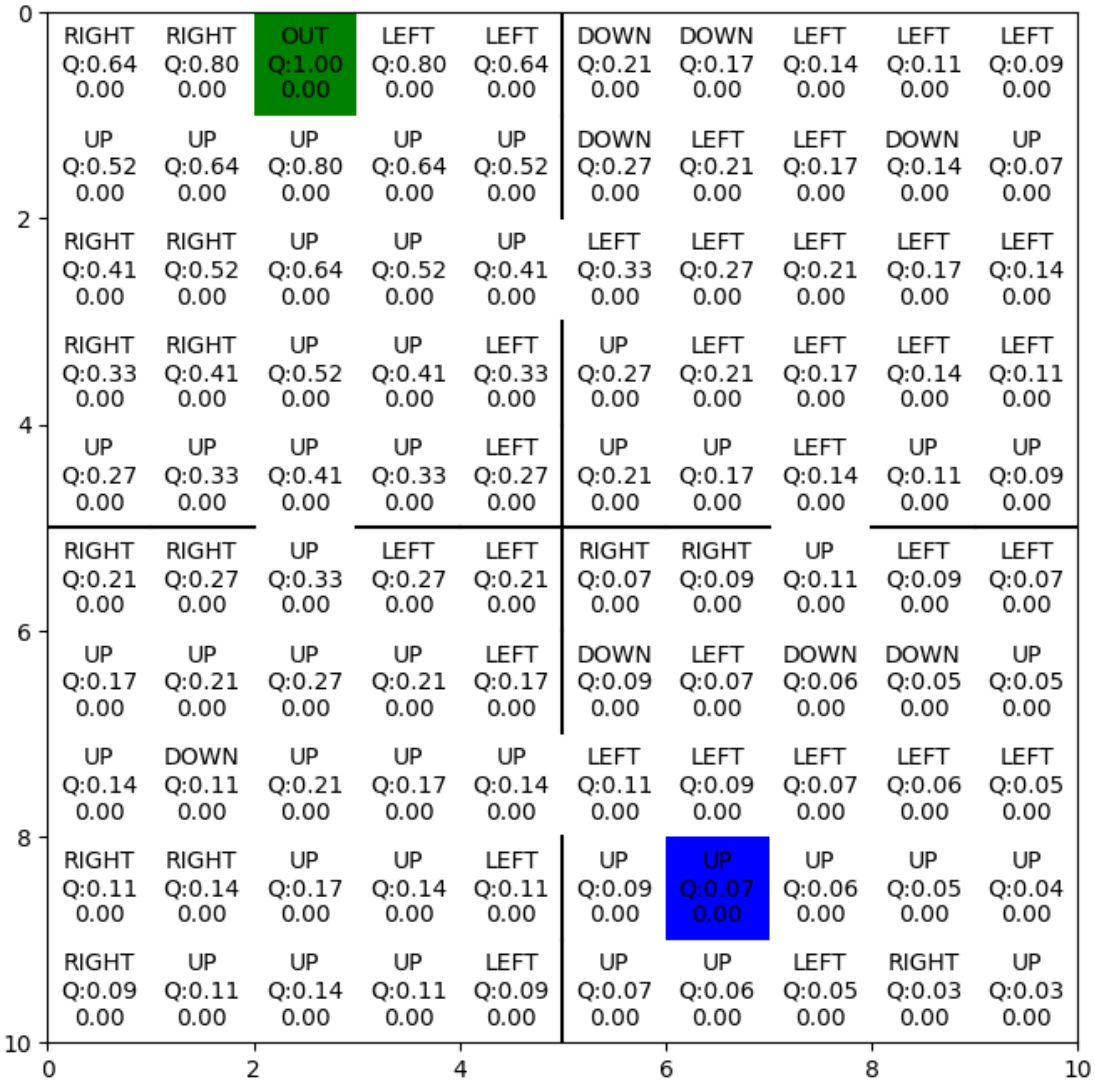
Como podemos observar, qlearning dio una buena política para el escenario de gridworld, evitando en lo posible a las trampas y haciendo que las políticas generen un camino que inevitablemente llegue al estado de salida. Sin embargo, también podemos apreciar que estas no son las políticas óptimas debido a que se puede llegar de forma más rápida al estado final en muchas de las casillas, principalmente en el sector izquierdo del mapa en donde evidenciamos más fácilmente esta situación, por lo que podemos asumir que el agente no tiene hyper parametros o recompensas óptimas para este escenario, o que realmente necesita más iteraciones para converger a la política más optima.

1.1.2 Laberinto

[ ]:
```
labyrinth_board, labyrinth_blocked_paths, labyrinth_objective, labyrinth_start␣
 ↪= generate_labyrinth()
labyrinth = GridBlockedPaths(
    labyrinth_board, labyrinth_objective, labyrinth_start,␣
 ↪labyrinth_blocked_paths)
```

```
q_learning = QLearning(labyrinth, random_seed, gamma=0.8)
print(
    f"El agente toma {q_learning.run()} muestras para converger valores a 3␣
 ↪decimales")
plot_scenario(q_learning)
```

```
El agente toma 17239 muestras para converger valores a 3 decimales
```



[ ]:
```
generate_q_table(q_learning)
```

[ ]:

	Estado	DOWN	LEFT	RIGHT	UP	OUT	Objetivo
0	(0, 0)	0.330134	0.354732	0.640000	0.425177	0.389066	No
1	(0, 1)	0.327680	0.279440	0.336384	0.512000	0.314786	No

2	(0, 2)	0.186931	0.293842	0.409600	0.206445	0.238680	No
3	(0, 3)	0.192395	0.209340	0.327680	0.272578	0.246982	No
4	(0, 4)	0.062472	0.124837	0.133693	0.262144	0.137566	No
5	(0, 5)	0.129439	0.156608	0.209715	0.154475	0.162297	No
6	(0, 6)	0.066322	0.088181	0.144982	0.167772	0.058121	No
7	(0, 7)	0.064242	0.005160	0.010555	0.134218	0.041186	No
8	(0, 8)	0.037891	0.052653	0.107374	0.053283	0.036366	No
9	(0, 9)	0.056046	0.041232	0.085899	0.033665	0.040178	No
10	(1, 0)	0.501834	0.510252	0.800000	0.636954	0.638167	No
11	(1, 1)	0.208896	0.238474	0.326400	0.640000	0.389069	No
12	(1, 2)	0.326029	0.293864	0.512000	0.497533	0.406235	No
13	(1, 3)	0.259558	0.260431	0.409600	0.408336	0.327062	No
14	(1, 4)	0.238023	0.204309	0.249004	0.327680	0.257956	No
15	(1, 5)	0.166933	0.167051	0.262144	0.209151	0.208459	No
16	(1, 6)	0.056756	0.097020	0.135524	0.209715	0.129723	No
17	(1, 7)	0.107374	0.059257	0.095720	0.050332	0.051053	No
18	(1, 8)	0.078796	0.073256	0.134218	0.069855	0.100535	No
19	(1, 9)	0.043111	0.050636	0.076496	0.107374	0.062034	No
20	(2, 0)	0.420480	0.420480	0.326400	0.607920	1.000000	Sí
21	(2, 1)	0.512000	0.512000	0.512000	0.800000	0.640000	No
22	(2, 2)	0.409600	0.409600	0.409600	0.640000	0.512000	No
23	(2, 3)	0.327680	0.327680	0.327680	0.512000	0.409600	No
24	(2, 4)	0.262144	0.262144	0.262144	0.409600	0.327680	No
25	(2, 5)	0.209715	0.209715	0.209715	0.327680	0.262144	No
26	(2, 6)	0.159656	0.166673	0.167772	0.262144	0.200359	No
27	(2, 7)	0.115642	0.063938	0.125287	0.209715	0.150459	No
28	(2, 8)	0.090311	0.090705	0.107374	0.167772	0.117536	No
29	(2, 9)	0.013152	0.053586	0.024802	0.134217	0.029851	No
30	(3, 0)	0.512000	0.800000	0.512000	0.640000	0.640000	No
31	(3, 1)	0.409600	0.640000	0.409600	0.640000	0.512000	No
32	(3, 2)	0.327680	0.511838	0.327680	0.512000	0.409600	No
33	(3, 3)	0.262144	0.408171	0.260614	0.409600	0.327680	No
34	(3, 4)	0.261518	0.320392	0.203326	0.327680	0.261719	No
35	(3, 5)	0.167772	0.262144	0.167772	0.209715	0.209715	No
36	(3, 6)	0.134218	0.209715	0.134218	0.209715	0.167772	No
37	(3, 7)	0.107373	0.166757	0.107374	0.167772	0.134217	No
38	(3, 8)	0.085891	0.100382	0.085893	0.134218	0.107343	No
39	(3, 9)	0.076174	0.071982	0.065306	0.107374	0.061502	No
40	(4, 0)	0.409600	0.640000	0.512000	0.512000	0.512000	No
41	(4, 1)	0.327680	0.512000	0.409600	0.512000	0.409600	No
42	(4, 2)	0.262144	0.409600	0.262144	0.409600	0.327680	No
43	(4, 3)	0.209715	0.327680	0.189491	0.293847	0.226544	No
44	(4, 4)	0.174468	0.262144	0.185041	0.078643	0.192444	No
45	(4, 5)	0.098090	0.209715	0.144422	0.135218	0.154587	No
46	(4, 6)	0.107374	0.167772	0.134218	0.167772	0.134218	No
47	(4, 7)	0.085899	0.134218	0.085899	0.134218	0.107374	No
48	(4, 8)	0.055471	0.107374	0.081169	0.092793	0.067186	No

49	(4, 9)	0.035836	0.085899	0.041502	0.061783	0.011521	No
50	(5, 0)	0.209715	0.081563	0.134218	0.031994	0.101347	No
51	(5, 1)	0.262144	0.209696	0.167765	0.162455	0.209701	No
52	(5, 2)	0.209715	0.327680	0.209715	0.209715	0.262144	No
53	(5, 3)	0.167768	0.209710	0.167772	0.262144	0.209715	No
54	(5, 4)	0.106819	0.139559	0.111660	0.209715	0.110226	No
55	(5, 5)	0.056792	0.045657	0.068719	0.047842	0.043191	No
56	(5, 6)	0.085899	0.052842	0.047711	0.054976	0.056690	No
57	(5, 7)	0.068719	0.107374	0.068719	0.068340	0.085899	No
58	(5, 8)	0.046121	0.059155	0.040292	0.085899	0.057604	No
59	(5, 9)	0.048508	0.036119	0.027681	0.068719	0.028879	No
60	(6, 0)	0.167772	0.151382	0.104341	0.133772	0.132570	No
61	(6, 1)	0.208452	0.209715	0.134205	0.134097	0.167726	No
62	(6, 2)	0.167772	0.262144	0.157964	0.158100	0.208757	No
63	(6, 3)	0.134209	0.209715	0.134212	0.209715	0.167771	No
64	(6, 4)	0.134181	0.167638	0.107124	0.167772	0.126480	No
65	(6, 5)	0.025448	0.052902	0.085899	0.063363	0.065282	No
66	(6, 6)	0.035047	0.068719	0.033421	0.035047	0.010555	No
67	(6, 7)	0.054953	0.085899	0.054970	0.053908	0.068719	No
68	(6, 8)	0.040495	0.064146	0.043980	0.068719	0.045395	No
69	(6, 9)	0.023037	0.041276	0.030156	0.054976	0.033819	No
70	(7, 0)	0.123164	0.134218	0.082412	0.054761	0.104334	No
71	(7, 1)	0.121230	0.167772	0.075745	0.065709	0.097617	No
72	(7, 2)	0.118422	0.209715	0.088149	0.111660	0.156608	No
73	(7, 3)	0.105251	0.167772	0.105888	0.148034	0.130426	No
74	(7, 4)	0.085827	0.134218	0.085066	0.133581	0.105888	No
75	(7, 5)	0.042676	0.067857	0.067948	0.107374	0.082652	No
76	(7, 6)	0.054976	0.023240	0.021254	0.028588	0.031419	No
77	(7, 7)	0.043980	0.068719	0.043977	0.043977	0.054974	No
78	(7, 8)	0.021242	0.030413	0.031045	0.054976	0.042088	No
79	(7, 9)	0.020459	0.043980	0.005404	0.028895	0.018901	No
80	(8, 0)	0.086510	0.107374	0.062890	0.080947	0.075793	No
81	(8, 1)	0.134218	0.000000	0.016452	0.085899	0.080950	No
82	(8, 2)	0.081543	0.167772	0.081593	0.054699	0.040265	No
83	(8, 3)	0.082932	0.134218	0.078756	0.101992	0.103024	No
84	(8, 4)	0.000000	0.032212	0.036797	0.107374	0.043809	No
85	(8, 5)	0.035184	0.085899	0.044526	0.049559	0.045957	No
86	(8, 6)	0.043980	0.038304	0.026673	0.026228	0.030845	No
87	(8, 7)	0.035182	0.054976	0.035179	0.035183	0.043933	No
88	(8, 8)	0.013689	0.036589	0.025807	0.043980	0.030968	No
89	(8, 9)	0.013959	0.000000	0.022518	0.017944	0.005404	No
90	(9, 0)	0.028038	0.085899	0.056163	0.063002	0.052220	No
91	(9, 1)	0.032212	0.064113	0.016493	0.068719	0.000000	No
92	(9, 2)	0.043801	0.134218	0.054741	0.016490	0.032212	No
93	(9, 3)	0.020616	0.107374	0.056436	0.083497	0.000000	No
94	(9, 4)	0.001526	0.029653	0.052965	0.085899	0.057128	No
95	(9, 5)	0.021376	0.068719	0.042435	0.010966	0.000000	No

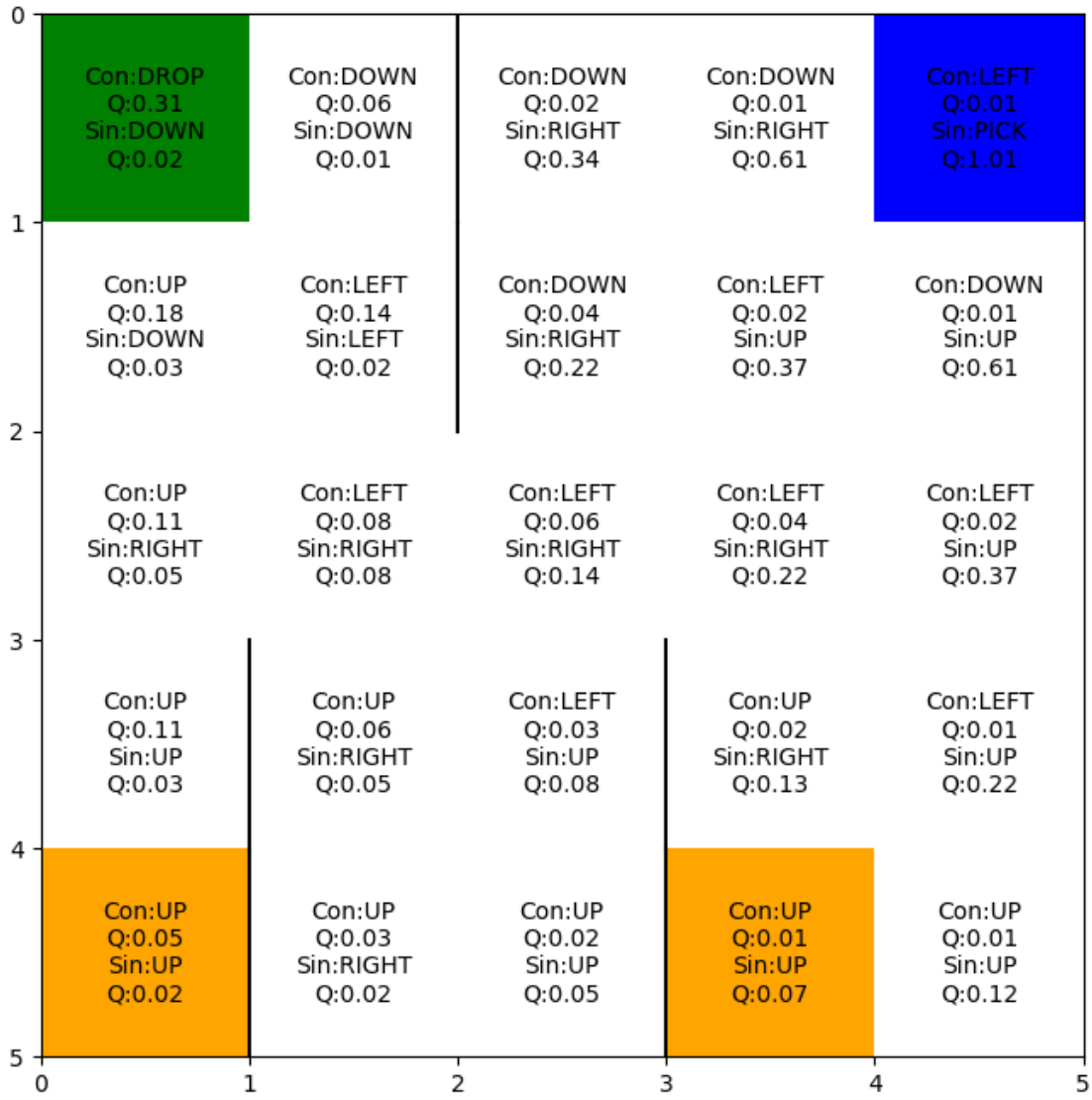
96	(9, 6)	0.035184	0.015283	0.012204	0.047573	0.012227	No
97	(9, 7)	0.028004	0.043980	0.035162	0.029837	0.034869	No
98	(9, 8)	0.019959	0.026857	0.023417	0.035184	0.027773	No
99	(9, 9)	0.006755	0.015867	0.019869	0.028147	0.021269	No

De igual forma que con el escenario de gridworld, podemos ver que, si bien las políticas siempre terminan llegando al objetivo, también podemos ver la falta de optimización en las mismas. No obstante, podemos apreciar que hay una menor cantidad de políticas no óptimas en este escenario, principalmente en el cuadrante más lejano, lo cual sigue concordando con el escenario de gridworld con políticas no óptimas a medida que se alejan de la recompensa, donde se espera que se encuentren menos muestras.

1.1.3 Taxi

[ ]:
```
taxi_board, taxi_blocked_paths, taxi_objective, taxi_start = generate_taxi()
taxi = TaxiGrid(taxi_board, taxi_objective, taxi_start, taxi_blocked_paths)
q_learning = QLearningTaxi(taxi, random_seed, alpha=0.1, epsilon=0.3)
print(
    f"El agente toma {q_learning.run()} muestras para converger valores a 3␣
 ↪decimales")
plot_scenario(q_learning)
```

```
El agente toma 5600 muestras para converger valores a 3 decimales
```



[ ]:
```
q_table = { "Estado": [],  }
for action in q_learning.environment.get_actions():
    q_table[action.name] = []
q_table["Estación"] = []
for state, passenger in sorted(q_learning.Q, key=lambda x: (x[0].x, x[0].y,␣
 ↪x[1])):
    if len(q_learning.Q[(state, passenger)]) < 2:
        continue
    q_table["Estación"].append("Sí" if state in q_learning.environment.
 ↪objectives else "No")
    q_table["Estado"].append(("Con pasajero" if passenger else "Sin pasajero")␣
 ↪+ f" ({state.x}, {state.y})")
```

```
    for action in q_learning.Q[(state, passenger)]:
        q_table[action.name].append(q_learning.Q[(state, passenger)][action])
DataFrame(q_table)
```

[ ]:

	Estado	DOWN	LEFT	RIGHT	UP	PICK	DROP	Estación
0	Sin pasajero (0, 0)	1.460042e-02	5.635938e-04	0.000000	0.000000	-2.913776	-4.094582	Sí
1	Con pasajero (0, 0)	1.276247e-01	5.110944e-02	0.050546	0.052772	-9.937783	0.306623	Sí
2	Sin pasajero (0, 1)	2.802304e-02	6.651662e-03	0.002534	0.004470	-9.930067	-9.948462	No
3	Con pasajero (0, 1)	6.653178e-02	1.047094e-01	0.059929	0.183688	-9.834779	-9.849925	No
4	Sin pasajero (0, 2)	5.807237e-03	1.496163e-02	0.046855	0.009493	-9.482687	-8.768309	No
5	Con pasajero (0, 2)	4.936976e-02	5.926110e-02	0.059138	0.114849	-9.920512	-9.904097	No
6	Sin pasajero (0, 3)	2.012613e-08	2.902554e-03	0.000000	0.027236	-7.709582	-6.125795	No
7	Con pasajero (0, 3)	1.919924e-02	3.118509e-02	0.033084	0.105278	-9.467714	-9.593144	No
8	Sin pasajero (0, 4)	4.697650e-04	2.851508e-07	0.000242	0.011352	-3.194404	-4.684297	Sí
9	Con pasajero (0, 4)	1.679541e-02	1.085672e-02	0.012375	0.049049	-9.825355	-7.343915	Sí
10	Sin pasajero (1, 0)	5.550310e-03	1.264775e-03	0.000375	0.000259	-3.439000	-3.439000	No
11	Con pasajero (1, 0)	5.888376e-02	5.875378e-02	0.057685	0.055473	-9.927420	-9.921161	No
12	Sin pasajero (1, 1)	8.906724e-03	1.514386e-02	0.002561	0.000637	-4.685369	-5.694450	No
13	Con pasajero (1, 1)	4.464206e-02	1.415356e-01	0.060716	0.059044	-9.929671	-9.934475	No
14	Sin pasajero (1, 2)	1.183975e-02	1.091270e-02	0.078124	0.001585	-9.730124	-9.896907	No
15	Con pasajero (1, 2)	2.883897e-02	7.865373e-02	0.039107	0.061986	-9.944162	-9.948002	No
16	Sin pasajero (1, 3)	4.648806e-04	1.673599e-03	0.045652	0.008878	-5.211047	-6.855224	No
17	Con pasajero (1, 3)	1.303196e-02	1.160417e-02	0.014376	0.058418	-9.955834	-9.962200	No
18	Sin pasajero (1, 4)	3.148821e-04	7.560428e-05	0.018379	-0.006156	-9.690968	-9.576088	No
19	Con pasajero (1, 4)	8.712299e-03	7.149261e-03	0.005006	0.034282	-9.681144	-9.742909	No
20	Sin pasajero (2, 0)	2.585600e-02	1.780873e-02	0.330643	0.038761			

```
-4.669824 -4.085824       No
21  Con pasajero (2, 0)  1.867874e-02  3.722297e-03  0.005048  0.004351
-6.510386 -8.900678       No
22  Sin pasajero (2, 1)  1.507611e-02  1.649680e-02  0.214091  0.018072
-5.664275 -5.682826       No
23  Con pasajero (2, 1)  3.955892e-02  1.566977e-02  0.009327  0.006128
-9.709501 -9.897321       No
24  Sin pasajero (2, 2)  2.310286e-02  3.837855e-02  0.130244  0.105862
-9.815861 -9.897864       No
25  Con pasajero (2, 2)  2.172134e-02  6.083085e-02  0.020930  0.018440
-9.973325 -9.966208       No
26  Sin pasajero (2, 3)  4.442021e-03  1.631413e-03  0.016004  0.078120
-9.769216 -9.257019       No
27  Con pasajero (2, 3)  8.025623e-03  3.195263e-02  0.013773  0.023518
-9.403706 -9.185593       No
28  Sin pasajero (2, 4)  5.287577e-03  8.745512e-04  0.000000  0.043994
-7.132021 -4.092834       No
29  Con pasajero (2, 4)  3.912143e-03  1.006864e-02  0.004009  0.018294
-9.970559 -9.960802       No
30  Sin pasajero (3, 0)  1.460116e-01  8.261876e-02  0.602709  0.231268
-9.587314 -9.659070       No
31  Con pasajero (3, 0)  9.851986e-03  6.544787e-03 -0.017032  0.003859
-9.896902 -9.946462       No
32  Sin pasajero (3, 1)  6.190357e-02  6.441910e-02  0.192182  0.361682
-6.507849 -7.451376       No
33  Con pasajero (3, 1)  1.554064e-02  1.824736e-02  0.006881  0.006372
-9.988054 -9.988205       No
34  Sin pasajero (3, 2)  5.650260e-02  6.341644e-02  0.217039  0.149203
-9.093701 -9.239980       No
35  Con pasajero (3, 2)  1.197781e-02  4.295292e-02  0.011962  0.011529
-9.979375 -9.980659       No
36  Sin pasajero (3, 3)  2.126278e-03  6.751056e-03  0.129889  0.052802
-9.008371 -9.819583       No
37  Con pasajero (3, 3) -6.490352e-02  9.124259e-03  0.005778  0.017676
-9.986014 -9.986117       No
38  Sin pasajero (3, 4) -5.191795e-02 -1.247440e-03  0.001330  0.065574
-4.154961 -7.764204       Sí
39  Con pasajero (3, 4)  5.334746e-03  4.385616e-03  0.001978  0.010513
-8.056666 -7.085156       Sí
40  Sin pasajero (4, 0)  3.573552e-01  3.509590e-01  0.580057  0.591321
1.003784 -9.429133        Sí
41  Con pasajero (4, 0)  5.893202e-03  5.949720e-03  0.003835  0.003943
-9.813268 -9.886948       Sí
42  Sin pasajero (4, 1)  1.802722e-01  2.061460e-01  0.345163  0.602513
-9.343106 -9.397085       No
43  Con pasajero (4, 1)  1.024115e-02  8.981555e-03  0.005388 -0.008239
-9.947353 -9.873417       No
```

```
44  Sin pasajero (4, 2)  1.177142e-01  1.190179e-01  0.179411  0.361609
-9.790511 -9.791108       No
45  Con pasajero (4, 2)  4.326718e-03  2.108517e-02  0.006515  0.004738
-9.988175 -9.987924       No
46  Sin pasajero (4, 3)  2.221977e-02  3.918929e-02  0.103363  0.217034
-7.868356 -7.871839       No
47  Con pasajero (4, 3) -4.853360e-02  1.060122e-02  0.003710  0.004940
-9.952207 -9.968797       No
48  Sin pasajero (4, 4)  1.176736e-02  3.480268e-03  0.006222  0.119924
-9.994363 -9.975751       No
49  Con pasajero (4, 4)  2.285124e-03 -2.780408e-01  0.001455  0.007329
-5.693533 -8.119650       No
```

En este escenario podemos ver un comportamiento que parece más aleatorio que los escenarios anteriores gracias a la naturaleza del problema. Sin embargo, analizando la qtabla y las políticas de la iteración final, podemos comprender que este comportamiento es lógico. Por un lado, las políticas favorecen la última iteración ya que las estaciones del pasajero y el objetivo son aleatorias, por lo que gracias a estos valores aleatorios que se contrarrestan entre sí y a los hyper parámetros usados, las últimas iteraciones tiene más peso sobre los qvalores, dando prioridad a las políticas resultantes. Por el otro lado, tenemos la información de la qtabla, la cual indica valores muy cercanos entre todas las acciones, siendo la excepción drop y pick, los cuales tienden a tener un valor negativo cercano a -10, debido a la gran penalización de ejecutar estas acciones de manera indebida y a la inhabilidad del agente de saber cuál es la estación del pasajero y objetivo para la iteración actual, siendo la excepción las estaciones involucradas en las últimas iteraciones por los motivos mencionados anteriormente.