2.1

Simpleton				
Eurística	Expandidos	Generados	Costo	Runtime
Zero	460	539	79	0.30
Manhattan	158	289	79	0.26
Euclidian	189	313	79	0.24
Octile	253	370	79	0.24

Barcelona				
Eurística	Expandidos	Generados	Costo	Runtime (seg)
Zero	14375	14436	3361	1.15
Manhattan	12367	12770	3361	1.03
Euclidian	12807	13082	3361	1.10
Octile	13182	13374	3361	1.09

Oficinas Chico				
Eurística	Expandidos	Generados	Costo	Runtime
Zero	30594	31216	636	1.71
Manhattan	3617	4248	636	0.87
Euclidian	6628	7309	636	0.88
Octile	10034	10692	636	1.19

Maze 4				
Eurística	Expandidos	Generados	Costo	Runtime (seg)
Zero	53704	54078	2236	2.36
Manhattan	35251	34753	2236	2.16
Euclidian	38311	39045	2236	2.03
Octile	41841	42479	2236	2.33

Starcraft				
Eurística	Expandidos	Generados	Costo	Runtime
Zero	927004	931163	4456	56.07
Manhattan	173719	167824	4456	12.01
Euclidian	295191	299341	4456	20.65
Octile	399291	403579	4456	13.98

La mejor heurística en este caso es la Manhattan, lo que resulta bastante lógico dado que hasta el momento sólo se ha implementado la conectividad 4. Esto significa que los autos sólo se mueven a las celdas colaterales y no hay movimientos diagonales. Luego, la distancia Manhattan, que corresponde a un $\Delta x + \Delta y$, es la más cercana a la realidad.

*Después de hacer todos estos tests, se corrigió el código de astar.py que venía malo.

2.2

Simpleton				
Algorítmo	Expandidos	Generados	Costo	Runtime
A*	162	292	79	0.26
Early-A*	186	321	79	0.23

Barcelona				
Algorítmo	Expandidos	Generados	Costo	Runtime
A*	12312	12770	3361	1.02
Early-A*	12357	12796	3361	1.10

Oficinas Chico				
Algorítmo	Expandidos	Generados	Costo	Runtime
A*	4153	4937	636	0.94
Early-A*	4632	5354	636	0.98

Maze 4				
Algorítmo	Expandidos	Generados	Costo	Runtime
A*	33691	34780	2236	1.98
Early-A*	33971	34968	2236	1.95

Starcraft				
Algorítmo	Expandidos	Generados	Costo	Runtime
A*	141903	152482	4456	4.76
Early-A*	191077	196157	4456	6.02

El algoritmo A* es mejor que el algoritmo Early-A*, dado que A* genera y expando menos nodos que Early-A*. La razón de fondo de este resultado es que Early-A*, al ir explorando los sucesores de los nodos en vez del nodo mismo, crea más nodos que después no va a valer la pena recorrer y que se van a quedar en Open.

2.3

Simpleton				
Algorítmo	Expandidos	Generados	Costo	Runtime
Connect-4	162	292	79	0.27
Connect-8	59	232	62.01	0.22

Barcelona				
Algorítmo	Expandidos	Generados	Costo	Runtime
Connect-4	12312	12770	3361	1.02
Connect-8	11773	12327	2791.03	1.26

Oficinas Chico				
Algorítmo	Expandidos	Generados	Costo	Runtime
Connect-4	4153	4937	636	0.94
Connect-8	2543	2855	588.35	0.68

Maze 4				
Algorítmo	Expandidos	Generados	Costo	Runtime
Connect-4	33691	34780	2236	1.98
Connect-8	29537	30367	1890.97	2.24

Starcraft				
Algorítmo	Expandidos	Generados	Costo	Runtime
Connect-4	141903	152482	4456	4.76
Connect-8	200132	119238	3704.44	9.22

Claramente la conectividad-8 le suma hartas cosas positivas a nuestra búsqueda. Lo primero es que los costos se reducen de manera notoria por el hecho de poder usar las diagonales. Luego, los nodos generados siempre son una cantidad menor con conectividad-8 y los nodos expandidos suelen ser de casi la mitad. Lo único que no se ve notoriamente favorable, es el Runtime, que en la medida que el mapa es más grande, va desfavoreciendo a la conectividad 8. Esto se debe a que, si bien, se expanden y generan menos nodos, los cálculos asociados a las diagonales se vuelven más complejos a causa de las raíces cuadradas y ahora hay más alternativas de movimiento (ahora hay 8 movimientos posibles), lo que hace que el árbol de decisiones sea más “ancho” y se tangan que analizar más opciones por cada nodo, lo que ralentiza aún más el proceso.

2.4

primitiveSimpleRight: 3 vecinos

Hacia adelante, doblar a la izquierda, doblar a la derecha.

primitiveSimpleDiagonal: 3 vecinos

Según la orientación del estado actual, tendrá un sucesor diagonal o un sucesor de movimiento recto, pero no ambos. A eso se le suman doblar en 45º a ambos lados.

primitiveFull: 5 vecinos

Según la orientación del estado actual, tendrá un sucesor diagonal o un sucesor de movimiento recto, pero no ambos. A eso se le suman doblar en 45º a ambos lados y doblar en 90º a ambos lados.

El costo del movimiento de giro en 90º se debe a que 2.5π es un cuarto del perímetro de una circunferencia de radio 5, que es justamente lo que está recorriendo el auto en esta curva.

2.5

Simpleton				
Algorítmo	Expandidos	Generados	Costo	Runtime
Connect-4	162	292	79	0.27
Connect-8	59	232	62.01	0.22
State Lattices	28	58	50.54	0.16

Barcelona				
Algorítmo	Expandidos	Generados	Costo	Runtime
Connect-4	12312	12770	3361	1.02
Connect-8	11773	12327	2791.03	1.26
State Lattices	188	231	214.85	0.08

Oficinas Grande				
Algorítmo	Expandidos	Generados	Costo	Runtime
Connect-4	164518	168867	2821	5.48
Connect-8	169305	136639	2340.07	7.72
State Lattices	916221	1014882	2293.07	135.62

Maze 4				
Algorítmo	Expandidos	Generados	Costo	Runtime
Connect-4	33691	34780	2236	1.98
Connect-8	29537	30367	1890.97	2.24
State Lattices	61291	65886	1515.21	6.23

Starcraft				
Algorítmo	Expandidos	Generados	Costo	Runtime
Connect-4	141903	152482	4456	4.76
Connect-8	200132	119238	3704.44	9.22
State Lattices	668150	677374	2296.4	108.82

Un detalle importante a considerar para esta pregunta, es que los problemas “aleatorios” que se le presentaban algoritmo State Lattices, comenzaban con una posición aleatoria, lo que significa que en varios caso los autos podían comenzar frente a una pared, sin tener el margen suficiente para dar la vuelta y seguir buscando el camino, sobre todo en los mapas más pequeños, como Simpleton y Oficinas_chico. El código se adaptó para que, cuando un problema no tuviera solución por esta causa, se hiciera un problema extra de manera que se completaran 5 problemas con solución por cada algoritmo. Esto a su vez significa que los problemas a los que se enfrentó State Lattices pueden ser distintos en algunos mapas a los que se enfrentaron Connect-4 y Connect-8.

El problema mencionado anteriormente se ve bien reflejado en los resultados de los mapas Simpleton y Barcelona, donde los mapas eran tan estrechos, que los problemas resueltos fueron con nodos que estaban muy cerca y cuya orientación inicial era ideal para lograr el objetivo. Un ejemplo de esto se ve en la imagen 1 de la pista de Barcelona. Esto obviamente significa que la cantidad de nodos abiertos, los costos y el runtime fue notoriamente menor.

Por otro lado, en los mapas grandes y con harto espacio para maniobrar, como los son Ofiinas Grandes, Maze 4 y Starcraft, se dio que la cantidad de nodos abierta fue ampliamente mayor. Esto se debe a que se están analizando una mayor cantidad de alternativas dado que hay muchas más formas de llegar a un mismo nodo (doblando primero, yendo en diagonal en cierto tramo, doblando después en 45º, etc.). El runtime también es bastante más alto, en parte porque se están abriendo más nodos y en parte porque las operaciones computacionales y los algoritmos utilizados son más complejos y requieren una mayor capacidad computacional. En lo que gana el modelo de State Lattices, es en encontrar un camino de menor costo, y esto se debe principalmente a que las curvas ahorran cierto costo respecto a lo que se tendría que recorrer estado por estado con los otros modelos.

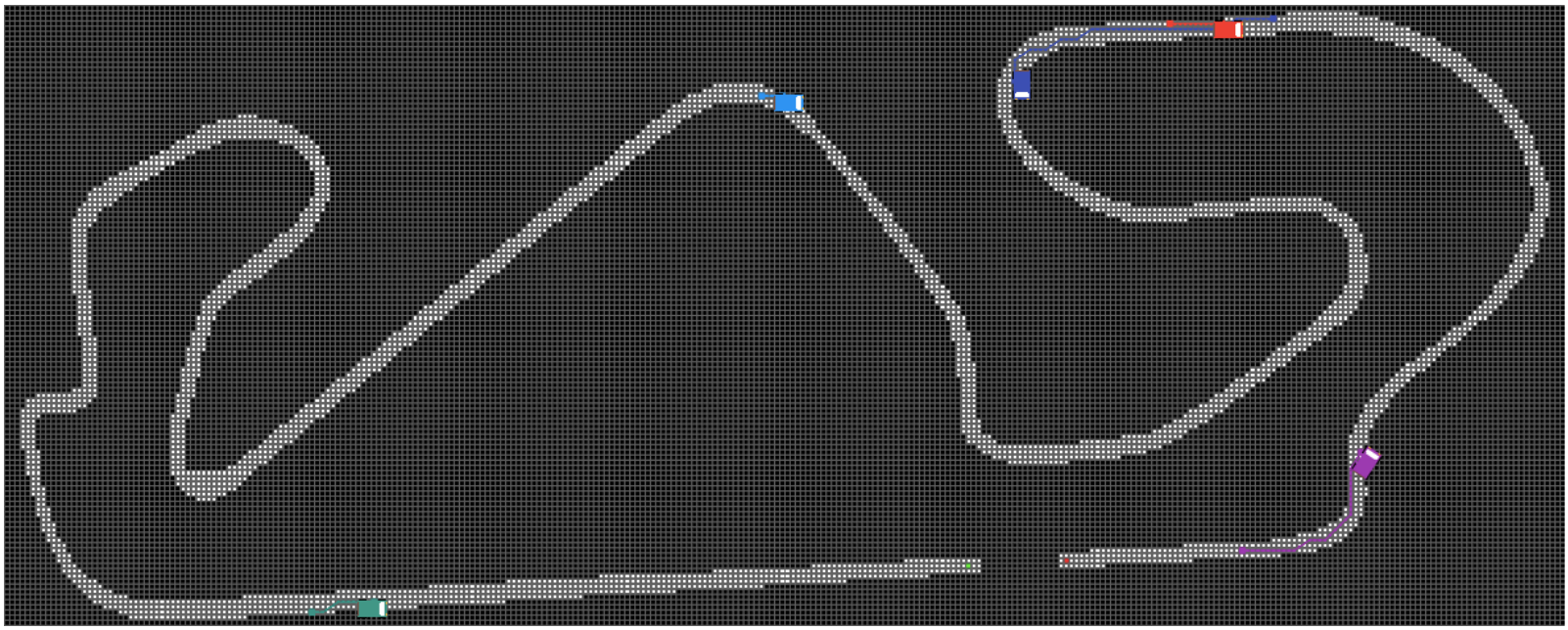

Imagen 1: State Lattices en Barcelona

2.6

El épsilon aumenta el costo de las curvas en 90º y 45º. Esto provoca que el algoritmo intente no doblar dentro de lo posible. Mientras mayor sea el épsilon, el algoritmo intentará evitar aún más las curvas. Al subir el costo, el algoritmo A* se demore más en extraer un nodo de curva de Open. El resultado final es que el algoritmo se demora más en encontrar una solución y abre más nodos en el camino.