Práctica 1: Desarrollo de agentes basado en técnicas de búsqueda heurística dentro del entorno GVGAI

Curso 2023-24
Tercer Curso de Ingeniería Informática

José Antonio Zamora Reyes

Técnicas de los Sistemas Inteligentes

UNIVERSIDAD DE GRANADA

TABLA DE RESULTADOS

Algoritmo	Mapa	Runtime (acumulado)	Tamaño de la ruta calculada	Nodos expandidos
Boulder Dash Maze (id 122)				
Dijkstra	Pequeño	46	710	11.4959
	Mediano	132	4593	29.14904
	Grande	934	22932	69,314065
A*	Pequeño			
	Mediano	----------No realizado----------		
	Grande			
RTA*	Pequeño	109	64	23.9916894
	Mediano	976	333	229.469263
	Grande	3927	1408	835.0293449
LRTA*	Pequeño	174	65	39.304404
	Mediano	3650	545	526.646603
	Grande	----------No funciona----------		
Boulder Dash Maze extended (id 123)				
RTA*	Pequeño	253	92	46.163347
	Mediano	976	333	209.0753769
	Grande	3942	1411	628.44833
LRTA*	Pequeño	303	91	44.185334
	Mediano	3650	545	511.793741
	Grande	----------No funciona----------		

PREGUNTAS

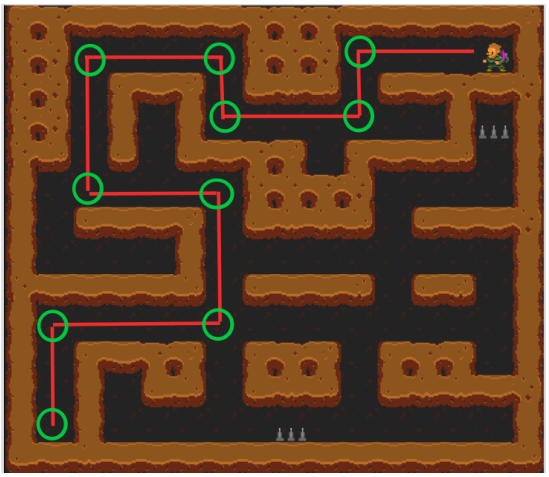
1. Si, en lugar de Boulder Dash Maze, operásemos con Labyrinth (juego 58 en GVGAI), ¿cuál sería la principal diferencia a nivel de representación de los estados? O, dicho de otra forma, si hubiésemos implementado estos mismos algoritmos (RTA*, LRTA*, A*, Dijkstra) para el juego Labyrinth, ¿estas implementaciones serían directamente aplicables a Boulder Dash Maze? Explore ambos juegos de forma comparativa e intenta dar una respuesta razonada.

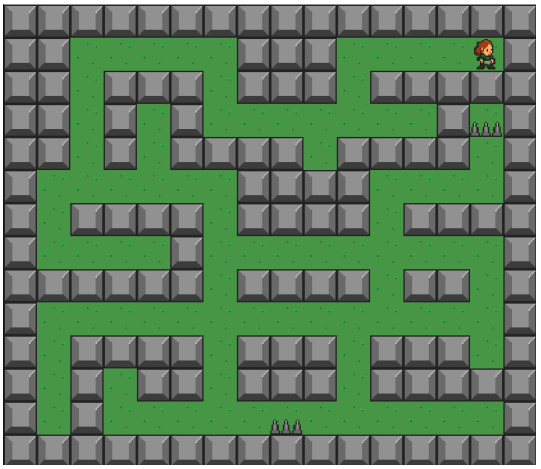
La principal diferencia entre ambos juegos , es que Labyrinth no utiliza el cambio de orientación , mientras que Boulder Dash Maze si la utiliza .
Esto nos indica que si hubiéramos implementado los algoritmos para el juego Labyrinth, estos mismos no pueden ser utilizados en el juego Boulder Dash Maze.

Explorando ambos juegos se puede observar claramente que el número de ticks aumenta.
En dijkstra y A* la orientación es fundamental en el momento de expandir nodos.
En RTA* y LRTA* se pueden expandir nodos o bien teniendo en cuenta la orientación o sin tenerla en cuenta (que los nodos solo sean casillas).
Aun haciéndolo sin tener en cuenta orientación en los nodos , los algoritmos de RTA* y LRTA* para Labyrinth , seguirán sin ser útiles para Boulder Dash Maze, debido a que en la actualización de la heurística para Labyrinth sólo se contemplaría

la heurística de la casilla a la que nos queremos mover más el coste de movimiento , mientras que para Boulder Dash Maze se contemplaría la heurística de la casilla a la que nos queremos mover más el coste de movimiento y más el coste de la rotación ,en el caso que se haya tenido que realizar.

Explorando los juegos podemos observar:
Voy a utilizar Dijkstra en el mapa más pequeño para verlo claramente:





Los mapas como se pueden observar son los mismos , sin embargo utilizando Dijkstra , nos da en Boulder Dash Maze un total de 46 tiks mientras que en Labyrinth nos dan 36 tiks.
La diferencia es de 10 tiks , que son el número de rotaciones que hemos realizado, (están marcadas con círculos verdes en las imágenes).

2. Imaginemos que implementamos dos versiones de RTA*: una, incorporando la orientación como parte de la representación de los estados, y otra sin incorporarla explícitamente (es decir, empleando un nodo por casilla e incrementando el coste dependiendo de hacia dónde nos movamos: +1 si la dirección del avatar coincide con la que queremos tomar; +2 si no es así). ¿De qué forma cambiaría el comportamiento del agente? ¿Cuál podríamos afirmar que es más correcto desde el punto del espíritu del propio algoritmo?

La principal diferencia en la actuación del agente , es que con orientación , este va a estar más tiempo en las casillas debido a que va a estar actualizando las heurísticas de aquellos nodos que corresponden a los cambios de orientación, estará actualizando heurísticas hasta que consiga que la heurística del nodo que produce el movimiento sea la menor de todas y pueda avanzar , sin embargo , si no usamos orientación el agente actuará de una manera más rápida y no tendrá que pararse en

una casilla y actualizar las heurísticas hasta que pueda avanzar , pudiendo explorar mucho más utilizando menos tiempo.
Implementado RTA* sin orientación , tomamos las casillas como nodos vecinos, de esta manera conseguiremos un mejor resultado que haciendo RTA* con orientación , esto se debe principalmente a que vamos a evitar movimientos que son realmente innecesarios (Cuando el agente entra en bucles de cambios de orientación actualizando heurísticas).
Como conclusión podemos decir que implementarlo sin orientación sería más correcto debido a que mejoramos en cuanto a eficiencia , pudiendo conseguir llegar al destino empleando menos tiempo.

3. ¿De qué forma habría que modificar los algoritmos de Dijkstra y A* para poder ser empleados en el nuevo juego Boulder Dash Maze extended? ¿Podrían ser empleados directamente sin realizar ninguna modificación a los algoritmos?

El algoritmo de Dijkstra y A* para funcionar en los mapas Boulder Dash Maze extend , tendrían que rediseñarse para que dentro del propio algoritmo, cuando encuentre una casilla de visibilidad, recalcule la ruta optima.

Explicado con un poco más de detalle , el algoritmo va a calcular una ruta hacia el portal , esta ruta pasará por una casilla de visibilidad , la subruta desde la casilla inicial a la casilla de visibilidad será óptima , sin embargo la subruta desde la casilla de visibilidad al portal no lo será , es por ello que debemos de actualizar la ruta óptima una vez que tengamos la nueva visibilidad del mapa.

Como indico anteriormente , los algoritmos tendrán que ser rediseñados por lo que no se pueden aplicar a estos mapas sin modificarlos ya que no tendrán la capacidad de conseguir llegar al portal.

4. ¿Se podría afirmar que RTA* es más eficiente que A*? ¿En qué casos, o bajo qué condiciones, preferiremos un algoritmo u otro?

Si hablamos de tiempo RTA* es más eficiente debido a que consigue llegar al portal de una manera más rápida que A* , sin embargo lo bueno de A* es que consigue el camino óptimo.

Si nuestro objetivo es llegar al portal en el menor tiempo posible , sin importar que encontremos o no el camino óptimo , el algoritmo que debemos utilizar es RTA* ya que este nos va a permitir explorar mapas más grandes dentro de las limitaciones de tiempo , es importante mencionar que si tenemos mapas en los que la visibilidad del agente es parcial , debemos utilizar también RTA* debido a que este se adapta a los cambios en el mapa.

En el caso en el que nuestro objetivo sea encontrar el camino óptimo , sin importar el tiempo , la mejor elección será A* , siempre y cuando las condiciones del mapa sean siempre las mismas y sean conocidas desde el inicio de la búsqueda.

Ejercicio 5

La principal diferencia entre los dos heatmaps es que en el de la izquierda , existe un cambio progresivo de la heurística entre casilla , mientras que en el de la derecha este cambio es bastante más brusco.

El heatmap de la izquierda corresponde al algoritmo LRTA* , esto nos lo indica el valor de las heurísticas. LRTA* actualiza con la menor heuristica de sus vecinos , esto va a provocar que haga más movimientos que RTA* , por lo que el valor de las heurísticas va a cambiar de una manera más progresiva , esto lo podemos observar en el heatmap ya que vemos que las heurísticas son menores en las casillas más próximas al portal y mayores en las más alejadas.

El heatmap de la derecha corresponde al algoritmo de RTA* que a diferencia de LRTA* , este actualiza con la segunda menor heurística de sus vecinos , esto provocará menos movimientos lo que conlleva que las heurísticas cambien de una manera mucho más brusca que en el heatmap de la izquierda.

Si generamos un heatmap de A* , será el heatmap de LRTA* el que más se asemejaría al heatmap de A* , debido a que LRTA* tras muchas ejecuciones tiene posibilidad de acercarse al óptimo.