PRÁCTICA 1 – TÉCNICAS DE LOS SISTEMAS INTELIGENTES

DESARROLLO DE UN AGENTE BASADO EN BÚSQUEDA HEURÍSTICA PARA EL ENTORNO GVGAI

APLICADO AL JUEGO BOULDER DASH

Antonio José Blánquez Pérez 3ºCSI
Grupo 2 – Viernes

En esta práctica se desarrolla un agente para el juego Boulder Dash, capaz de reaccionar en cinco situaciones distintas: ir desde la posición inicial al portal de salida (comportamiento deliberativo simple), ir desde la posición inicial hasta el portal de salida recogiendo 10 gemas en el proceso (comportamiento deliberativo compuesto), aguantar 2000 ticks evitando un enemigo (comportamiento reactivo simple), aguantar 2000 ticks evitando dos enemigos(comportamiento reactivo compuesto) y recoger 10 gemas e ir al portal de salida evitando a un enemigo (comportamiento reactivo-deliverativo). A partir de aquí se van a detallar tanto las técnicas y heurísticas que se han utilizado para resolver estas cinco situaciones correctamente como algunas reflexiones sobre la bondad de estas últimas.

NIVEL 1: COMPORTAMIENTO DELIBERATIVO SIMPLE

En este primer nivel se pide tan solo ir desde la posición inicial hasta el portal de salida, para lo que se ha implementado una versión de A* aplicada a los mapas de Boulder Dash. Su misión es encontrar el camino óptimo entre los dos puntos, y se ha elegido este algoritmo porque, siendo más simple de implementar que sus extensiones, es suficientemente rápido y eficaz para la extensión y complejidad de los mapas del problema que nos ocupa.

Para implementar A* es esencial identificar la función f(n) = g(n) + h(n) que identifica un nodo, tal que g es la función que indica el coste desde la posición inicial hasta el nodo en cuestión y h una estimación del coste restante desde el nodo hasta la solución. En este juego el mapa es cuadriculado, es decir, cada posición posible es una coordenada de la cuadrícula, y los movimientos solo pueden efectuarse a una de las cuatro cuadrículas colindantes, por lo que las acciones de movimiento serán arriba, abajo, izquierda y derecha. Es necesario tener en consideración también que el movimiento funciona según la orientación que tenga el personaje, de tal manera que si anteriormente se ha movido a la derecha, estará orientado a la derecha, así que si ahora se quisiera mover hacia arriba, el primer input hacia arriba cambiará la orientación hacia arriba y el segundo realizará el movimiento. Esta condición afecta de manera decisiva en el coste de un camino, así se ha implementado esta sumando, por cada movimiento, uno a g si el movimiento anterior había sido en la misma dirección y dos si había sido en otra.

Para la función h se ha elegido calcular la distancia Manhattan hasta el objetivo, ya que se ajusta más al coste real que, por ejemplo, una distancia euclídea, debido a la geometría del mapa como se ha explicado anteriormente. Podemos decir que esta heurística es admisible porque siempre nos va a dar un valor más pequeño que el coste real, $h(n)<h^*(n)$, debido a que el camino más corto hasta el objetivo sería este cálculo y que hay que tener en cuenta que los giros tienen el doble de coste y que puede haber obstáculos en medio.

Para la implementación se han usado, para las listas de abiertos y cerrados, Arraylist del tipo Node proporcionado por el propio GVGAI ya que una primera implementación propia ha dado multitud de problemas. El coste extra se ha establecido de tal manera que, al expandir cada nodo este tendrá cuatro hijos, tres de ellos con un coste de dos sumado al coste del nodo anterior y el otro, el que se da en la misma dirección del movimiento anterior, con tan solo un uno sumado al coste(teniendo en cuenta obviamente la orientación inicial para la primera iteración). Además, para controlar el camino final se ha usado el campo parent de Node para guardar el padre de cada nodo, de manera que al terminar reconstruimos el camino que ha llevado al nodo solución y lo reparamos duplicando cada input de movimiento si el anterior no es el mismo.

Con todo esto conseguimos el resultado esperable, obteniendo el camino hasta la solución evitando los obstáculos del mapa y superando por tanto este nivel.

NIVEL 2: COMPORTAMIENTO DELIBERATIVO COMPUESTO

El segundo nivel es tan sólo una extensión del primero, debiendo establecer un mecanismo deliberativo para recoger 10 gemas antes de llegar al portal. Para abordar este problema nos serviremos del algoritmo A* implementado en el nivel anterior, estableciendo como objetivo la gema más cercana a nuestra posición en cada caso hasta llegar a 10, entonces se buscar el camino al portal hasta terminar.

Este comportamiento es de una tipología greedy que selecciona la distancia Manhattan más corta hasta la primera gema que recogeremos. De esta manera no tenemos por que obtener el camino óptimo, ya que puede haber muros entre nosotros y la gema que seleccionemos que no tengamos en cuenta de manera que el camino sea más largo de los esperado; y aunque no los hubiera tampoco obtendríamos el camino óptimo. Si encontramos un muro también es posible coger otra gema por el camino, hecho que también se ha tenido que tener en cuenta para no coger más gemas de las necesarias.

Sin embargo, aunque no sea óptimo, es un procedimiento que funciona en todos los casos(salvo en algunas excepciones como tener gemas inalcanzables que no se han abordado en esta práctica) y es válido para el objetivo propuesto, obteniendo así resultados satisfactorios en este nivel y no demasiado lejanos del óptimo.

NIVEL 3: COMPORTAMIENTO REACTIVO SIMPLE

En este tercer nivel se ha implementado una heurística para evadir los ataques de un enemigo, la cual es simple pero efectiva. Lo que se ha hecho es elegir en cada iteración el input que lleve a una posición cuya distancia Manhattan al enemigo sea la mayor.

De esta manera se consigue un algoritmo muy simple pero que proporciona un resultado bastante bueno. El personaje suele irse a las esquinas que es donde la mayor distancia consigue con el enemigo, de tal forma que cuando el enemigo se acerca en la y cambia a la esquina superior, y cuando se acerca en la x cambia a la otra esquina lateral. Este cambio se produce al restringir que si alguno de los inputs lo intenta mover hacia un muro, este input no se puede elegir en esa iteración, lo que también provoca un cambio de posición constante en las esquinas.

Debido a que el problema tiene una naturaleza aleatoria, dado que el enemigo se mueve aleatoriamente, se ha hecho estudio obteniendo los resultados de 1000 iteraciones, obteniendo un win-rate de 99.5%. El otro 5% el enemigo acaba con nuestro personaje.

Las razón por las que el enemigo puede llegar a nosotros es básicamente que, en una esquina, si el enemigo se acerca por la diagonal del rectángulo que es el mapa, el personaje no se decide por ninguna de las dos direcciones de huida posibles. Esto último se debería de solucionar en el momento en el que el enemigo se inclina hacia un lado(recordad que no es posible moverse en diagonal en el mundo del juego) para atacarnos, pero como el escorpión no necesita establecer la orientación para moverse nos caza antes de poder salir.

A pesar de haber casos en los que el enemigo nos caza podemos concluir que un 99.5% es un gran resultado y que, por tanto, el comportamiento resultante cumple perfectamente los objetivos planteados para este nivel.

NIVEL 4: COMPORTAMIENTO REACTIVO COMPUESTO

Para el cuarto nivel se nos pide manejar la misma situación que en el caso anterior pero con dos enemigos. Este caso es más complejo que el anterior, ya que manejar ambos enemigos no es demasiado fácil. Tras probar varias heurísticas, he decidido seleccionar una extensión de la anterior, ya que tras estudiarlas es la que mejor resultado ha dado, de manera que se ha adaptado la idea de escoger el input que proporcione mayor distancia al enemigo al hecho de enfrentar dos.

Es importante estudiar como relacionar la distancia a ambos enemigos para obtener el mejor resultado posible, siendo la primera idea que se nos podría ocurrir la de sumar ambas distancias. Sin embargo, se ha elegido multiplicar ambas distancia, ya que al sumar tenemos el problema de que obtenemos el mismo resultado con ambos enemigos a una media distancia que con uno lejos y otro cerca de nosotros. Es por ello que se ha seleccionado la multiplicación, ya que nos premia tener a los dos enemigos a una distancia controlada antes que tener a un enemigo muy lejos y a otro cercano a nosotros con el consiguiente peligro.

Dicho ésto se ha visto necesario realizar el mismo estudio que en el nivel anterior, obteniendo un 80.7% de win-rate. No es un resultado muy espléndido, pero de todas las ideas es esta la que ha obtenido el mejor resultado; siendo las razones del 19.3% restante una extensión de las que provocaban el 0.5% del nivel anterior. Los escorpiones atacan ambos desde la diagonal, una estando cerca y otro lejos, y el más cercano no ataca antes de que podamos salir debido a que no necesita establecer la orientación de su movimiento.

Aunque la proporción de victorias sea peor que en el caso anterior(resultado lógico por otra parte) considero que es un porcentaje aceptable dada la complejidad del problema y que lo resuelve en la mayor parte de los casos.

NIVEL 5: COMPORTAMIENTO REACTIVO-DELIBERATIVO

En este quinto y último nivel se pide un híbrido de los comportamiento reactivos y deliberativos de los niveles anteriores, de tal manera que debemos recoger 10 gemas y terminar llegando al portal de salida esquivando a un enemigo.

Para ello se ha hecho uso de ambas heurísticas utilizadas en los niveles anteriores, tanto A* como la de maximizar distancias, de manera conjunta para resolver el problema. Tan solo es necesario plantear el problema del nivel 2 para recoger las gemas, estableciendo un margen de acción para el que se deja de buscar gemas y se empieza a usar la heurística del nivel 3 con la intención de evitar al enemigo, para posteriormente recalcular una ruta hacia la gema más cercana. En este caso el margen se ha establecido a una distancia Manhattan de 5, que ha sido el que mejor resultado a dado de los que se han provocado. Un margen demasiado bajo haría que el enemigo nos pudiera coger antes de escapar de él, pero un margen demasiado alto nos haría perder la referencia de las gemas demasiadas veces sin un peligro inminente perdiendo eficiencia por ello.

Se ha hecho un estudio de 500 ejecuciones del problema, obteniendo unos grandes resultados del 99.8% de win-rate y una media de unos 140 ticks, de manera que los resultados son altamente satisfactorios y claramente válidos para el problema planteado