INTELIGENCIA ARTIFICIAL

CURSO 2022-23

PRACTICA 2: Repertorio de preguntas para la autoevaluación de la práctica 2.

APELLIDOS Y NOMBRE	Gómez López, Javier		
GRUPO TEORÍA	DGIIM	**GRUPO PRÁCTICAS**	A1D

Instrucciones iniciales

En este formulario aparecen preguntas que requieren breves explicaciones relativas a cómo el estudiante ha hecho algunas partes de esa implementación y qué cosas ha tenido en cuenta.

Enumera los niveles presentados en su práctica (Nivel 0, Nivel 1, Nivel 2, Nivel 3, Nivel 4):

Nivel 0, Nivel 1, Nivel 2 y Nivel 3

Nivel 1-Anchura para el agente sonámbulo

(a) ¿En qué se diferencia desde el punto de vista de la implementación el algoritmo que has usado en este nivel en relación al del nivel 0? (enumera los cambios y describe brevemente cada uno de ellos)

Este caso es muy similar al del nivel 0. La implementación varía en este caso porque tenemos que tener en cuenta los movimientos del sonámbulo, que son los que al final nos pueden dar un estado solución. Las principales diferencias son las siguientes:
- Creamos un nuevo método llamado VeoSonámbulo(stateN0 &st) que, dado un estado, nos dice si el jugador puede ver al sonámbulo desde su situación o no. Es importante destacar que dada la naturaleza del estado que definimos en el nivel 0, no ha sido necesario crear un estado nuevo para este nivel.
- Si vemos al sonámbulo, generamos primero los nodos hijos asociados a los movimientos del sonámbulo. El primero es el que mueve al sonámbulo hacia delante, y tras generarlos comprobamos que este es solución. De serlo, el algoritmo termina y hemos llegado a una solución. De no ser solución, seguimos generando el resto de hijos con los restantes movimientos del sonámbulo y los movimientos del jugador.
- Hemos tenido que ampliar el operador < para los nodos de este nivel, teniendo en cuenta también los atributos relativos al sonámbulo.

En general, es un algoritmo de búsqueda en anchura igual que el del nivel 0, pero teniendo en cuenta que el estado solución es distinto al primero y que hay que generar más hijos al tener en cuenta más movimientos.

Nivel 2-Dijkstra para el agente jugador

(a) ¿Qué es propio de este nivel que no tuviste que tener en cuenta en los niveles anteriores? (enumera los cambios y describe brevemente cada uno de ellos y que han implicado en la implementación)

Este nivel ya cambia radicalmente el planteamiento respecto a los dos anteriores. En este caso tenemos que implementar una búsqueda por coste uniforme. Puesto que aquí ya tenemos que tener en cuenta el coste de la batería, tenemos que tener en cuenta más factores como el bikini o las zapatillas del jugador. Por ello, tenemos que ampliar el estado de los niveles anteriores a uno que tenga dos variables que nos digan si el jugador tiene bikini o zapatillas, además de ampliar el operador. También hemos ampliado el nodo, donde también tenemos ahora una variable que nos indica el coste acumulado en ese nodo. Las principales diferencias en la implementación son las siguientes:
- Los nodos a explorar deben de seguir el algoritmo de Dijkstra, y por ello no podemos sacarlos de la lista de abiertos en orden de inserción, si no que tenemos que seguir un orden, en este caso el de menor coste acumulado. Por ello, en lugar de una lista de nodos abiertos, hemos usado una priority_queue.
- También, por simplicidad de cara a este nivel y al siguiente, hemos sustituido nuestro set de nodos por un set de estados para los cerrados.
- Hemos creado una función CalcularCoste(), que nos da el coste de realizar una determinada acción desde una determinada posición.
- Primero generamos todos los hijos posibles a partir de un nodo, en este nivel solo teniendo en cuenta los movimientos del jugador. Una vez generamos todos los nodos hijos, pasamos a ir sacando nodos de nuestra cola de abiertos, y comprobando que no los hemos explorado ya. Cuando tenemos un nodo que hemos extraído de abiertos y no está en los explorados, ahora pasamos a comprobar si es solución.

Es un enfoque muy distinto a los dos niveles anteriores, por la naturaleza del algoritmo y por el hecho de tener que comprobar la solución una vez generados todos los abiertos nuevos.

(b) ¿Has incluido dentro del algoritmo de búsqueda usado en este nivel que si pasas por una casilla que da las zapatillas o el bikini, considere en todos los estados descendientes de él el sonámbulo o el jugador tiene las zapatillas y/o el bikini? En caso afirmativo, explicar brevemente cómo.

Puesto que en nuestro stateN2 guardamos la información de si tenemos zapatillas o bikini, nuestra función CalcularCoste() tiene en cuenta si tenemos alguno de estos objetos a la hora de devolvernos el coste de una determinada acción. Por tanto, al acumular este coste en el nodo, tendrá una posición distinta en nuestra priority_queue a la que tendrá el nodo que realiza esa acción sin el determinado objeto. Por tanto, sí se tiene en cuenta.

Nivel 3-A* para el agente sonámbulo

(a) ¿Qué diferencia este algoritmo del de Dijsktra que tuviste que implementar en el nivel anterior? (enumera los cambios y describe brevemente cada uno de ellos y que han implicado en la implementación)

En este caso vemos bastantes similitudes en el salto del nivel 0 al 1 con el salto del nivel 2 al nivel 3. Mantenemos los cambios mencionados en el nivel anterior, pero pasamos a ampliar nuestro nodo y nuestro estado.
Ahora tenemos que realizar también acciones con el sonámbulo, por tanto, también tenemos que guardar en nuestro estado la información acerca de los objetos que puede portar el sonámbulo.

Además, en nuestro nodo, pasamos a tener en cuenta no solo el coste, si no la heurística de ese nodo y la suma de ambos (que es el dato que realmente nos influye). Las principales diferencias en la implementación son las siguientes:
- Al igual que en el nivel 1, en caso de ver al sonámbulo, generamos todos los posibles hijos de ese nodo con las acciones del sonámbulo, y tras esto, los de las acciones del jugador. En caso de no ver al sonámbulo, se generan los nodos de los movimientos del jugador.
- Ahora, al generar cada hijo, además de usar nuestra función CalcularCoste() para ver el coste de la acción, calculamos la heurística desde el nuevo estado y sumamos. El coste es acumulado, pero la heurística no es acumulativa para cada nodo.
- La función CalcularCoste() ahora también tiene en cuenta las acciones del jugador, y si este tiene o no objeto (bikini o zapatillas).
- Tenemos una nueva función distanciaChebyshev() que calcula la distancia de Chebyshev de dos ubicaciones dadas.

(b) Describe la heurística utilizada para resolver el problema

En nuestro caso, la heurística elegida es la distancia de Chebyshev, o norma del máximo, desde el sonámbulo hasta el punto objetivo. La distancia de Chebyshev calcula el máximo entre la diferencia de la posición de la fila del sonámbulo con la fila del objetivo (en valor absoluto) y la diferencia de la posición de la columna del sonámbulo con la columna del objetivo (en valor absoluto). Esta heurística es la que usamos siempre, independientemente del hijo que se esté generando. Es claro que es una heurística aceptable, pues nunca va a ser mayor del coste real.

Somos conscientes de que no es la mejor heurística posible (vemos que tarda mucho), pero sí la que hemos sido capaces de implementar con éxito. Otra heurística que podríamos considerar más efectiva sería usar la distancia Manhattan con el jugador hasta la casilla más cercana en la que ve al sonámbulo, y mantener la de Chebyshev como es usada ahora mismo con el sonámbulo.

Nivel 4-Reto (Max. Puntuación en misiones)

(a) Haz una descripción general de tu estrategia general con la que has abordado este nivel. Indica bajo qué criterios es el jugador o el sonámbulo el que va al objetivo. Explica brevemente las razones de esos criterios.

(b) ¿Qué algoritmo o algoritmos de búsqueda usas en el nivel 4? Explica brevemente la razón de tu elección.

(c) ¿Bajo qué condiciones replanifica tu agente?

(d) Explica el valor que le has dado a la casilla desconocida en la construcción de planes cuando el mapa contiene casillas aún sin conocer. Justifica ese valor.

(e) ¿Has tenido en cuenta la recarga de batería? En caso afirmativo, describe la política usada por tu agente para proceder a recargar.

(f) ¿Has tenido en cuenta la existencia de aldeanos y lobos para definir el comportamiento del agente? En caso afirmativo, describe en qué sentido los has tenido en cuenta.

(g) Añade aquí todas los comentarios que desees sobre el trabajo que has desarrollado sobre este nivel, qué consideras que son importantes para evaluar el grado en el que te has implicado en la práctica y que no se puede deducir de la contestación a las preguntas anteriores.

Comentario final

Consigna aquí cualquier tema que creas que es de relevancia para la evaluación de tu práctica o que quieras hacer saber al profesor.

Me ha parecido una práctica igualmente interesante que la anterior, pero está he sentido más seguridad a la hora de ver que estaba realizando correctamente los ejercicios pedidos. También quiero destacar vuestra labor como profesores, pues considero que está siendo de las mejores que he tenido en la carrera. Sois muy atentos, muy rápidos contestando nuestras dudas y muy cercanos. De verdad, aunque no sea la última práctica, mil gracias por todo.