

INTELIGENCIA ARTIFICIAL

CURSO 2024-25

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

APELLIDOS Y NOMBRE	Olivares Martos, Arturo		
GRUPO TEORÍA	DGIIM	GRUPO PRÁCTICAS	DG2

Instrucciones iniciales

En este formulario se proponen preguntas que tienen que ver con ejecuciones concretas del software desarrollado por los estudiantes. También 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.

En las preguntas relativas al funcionamiento del software del alumno, estas se expresan haciendo uso de la versión de invocación en línea de comandos cuya sintaxis se puede consultar en el guion de la práctica.

El estudiante debe poner en los recuadros la información que se solicita.

En los casos que se solicita una captura de pantalla (**ScreenShot**), extraer la imagen de la ejecución concreta pedida (sustituyendo la llamada practica25G por practica2). En los **niveles 0 y 1**, esta captura será de la situación final de la simulación en el modo "mapa" en la que se ve lo que los agentes descubrieron. En los **niveles 2 y 3** donde aparezca la línea de puntos que marca el recorrido (justo en el instante en el que se construye obtiene el plan). Además, en dicha captura debe aparecer al menos el nombre del alumno. Ejemplos de imágenes se pueden encontrar en [Imagen1](#) y en [Imagen2](#).

Consideraciones importantes:

- Antes de empezar a rellenar el cuestionario, **actualiza el código de la práctica con los cambios más recientes**. Recuerda que puedes hacerlo o bien realizando **git pull upstream main** si has seguido las instrucciones para enlazar el repositorio con el de la asignatura, o bien descargando desde el enlace de GitHub el zip correspondiente, y sustituyendo los ficheros **rescatador.cpp**, **rescatador.hpp**, **auxiliar.cpp** y **auxiliar.hpp** por los vuestros.
- Si en alguna ejecución consideras que tu agente se ha visto perjudicado puedes añadirlo a los comentarios en el comentario final (al final del documento).

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

Todos. Nivel 0, Nivel 1, Nivel 2, Nivel 3 y Nivel 4

Nivel 0-El Despertar Reactivo

(a) Rellena los datos de la tabla con el resultado de aplicar


`./practica2SG ./mapas/mapa30.map 0 0 24 10 2 17 17 0 3 3 0`

<p>Screen Shot</p>	
<p>Instantes de simulación no consumidos</p>	<p>2968</p>
<p>Coste de Energía (Rescatador)</p>	<p>32</p>
<p>Coste de Energía (Auxiliar)</p>	<p>15</p>

(b) Rellena los datos de la tabla con el resultado de aplicar

`./practica2SG ./mapas/mapa30.map 0 0 16 9 2 16 14 6 3 3 0`

Screen Shot



```
arturoolvrs@arturoolvrs-MacBookAir:~/Desktop/Practica2_IA$ echo "Arturo Olivares Martos"
Arturo Olivares Martos
arturoolvrs@arturoolvrs-MacBookAir:~/Desktop/Practica2_IA$ ./practica2 ./mapas/mapa30.map 0
0 16 9 2 16 14 6 3 3 0
```

Instantes de simulación no consumidos	2960
Coste de Energía (Rescatador)	25
Coste de Energía (Auxiliar)	40

(c) Rellena los datos de la tabla con el resultado de aplicar
`./practica2SG ./mapas/gemini2.map 0 0 3 10 2 3 13 6 3 3 0`

Instantes de simulación no consumidos	2715
Coste de Energía (Rescatador)	123
Coste de Energía (Auxiliar)	285

Nivel 1-La cartografía de lo Desconocido

- (a) Describe brevemente cuál es el comportamiento que has implementado en los agentes para explorar el mundo. Indica si has usado los dos. Si has usado los dos indica describe las diferencias que hubiera entre ellos. (enumera los cambios y describe brevemente cada uno de ellos)

Sí, he empleado ambos agentes y el funcionamiento de ambos es análogo (evidentemente con las limitaciones de movimientos y acceso a casillas de cada uno).

En cada iteración, en primer lugar se actualizan las variables de estado de los agentes. Se actualizan los mapas Resultado, Cotas y Entidades, y también la variable *tieneZapatillas*. Además, empleo dos matrices. La primera almacena el número de veces que el agente ha visto por el sensor una casilla; mientras que la segunda contabiliza el número de veces que el agente la ha visitado. Estas matrices también son actualizadas en este momento.

Posteriormente, se decide la acción a tomar. Si el agente ya se encontraba en un movimiento que requiera más de una acción (por ejemplo, un giroSL en el auxiliar) se lleva a cabo la acción correspondiente. En caso contrario, se decide a qué casilla ir, y en función de eso se determina qué acción tomar. Las casillas que se contemplan son las que el agente tiene constancia de que puede llegar. Estas son las casillas accesibles entre las siguientes:

- 1,2,3 y las dos inmediatamente de los lados (para el auxiliar). Suponiendo que el agente tiene orientación norte, estas serían:

1	2	3
-2	^	-4

- 1,2,3,4,6,8 y para cada lado, las que se llegarían andando y corriendo. Suponiendo que el agente tiene orientación norte, estas serían:

4	6	8		
1	2	3		
-1	-2	^	-4	-5

Notemos que se consideran las de los laterales para permitir movimientos en L, por ejemplo (imaginemos que hay unas zapatillas en la casilla 5). No obstante, nunca se plantea ir hacia atrás, puesto que estas casillas ya se habrán planteado anteriormente.

De las casillas mencionadas (en adelante alcanzables, que no accesibles), se descartan aquellas que no son accesibles para el agente, ya sea por que el otro agente está ahí, o porque no puede acceder por altura o por el tipo de terreno. También se descartan aquellas casillas por las que no le permitimos ir. Puesto que se afirmó que la gran mayoría de senderos y caminos estaban conectados, tampoco se le permite ir por caminos, senderos, puestos base y zapatillas. Las casillas alcanzables que son accesibles se denominarán accesibles.

De entre las accesibles accesibles, si se ve una casilla de zapatillas y el agente aún no las posee, va directamente a ellas. Si ya las posee o no ve una casilla de este tipo, continúa el proceso.

A continuación, se toma la casilla que tenga **menor** puntuación, siguiendo el siguiente criterio:

- $Puntuacion = PESO_VISTA * vecesVista + PESO_VISITADA * vecesVisitada$

donde el número de veces vistas y visitadas se obtienen de las matrices anteriormente mencionadas y los factores PESO_VISTA y PESO_VISITADA son constantes especificadas por el programador. Haciendo pruebas determiné que PESO_VISTA=0.6 y

PESO_VISITADA=1 eran valores válidos para un buen resultado, pero evidentemente no puedo

garantizar que sean los mejores. Seguramente con más tiempo, más pruebas, y más ordenadores podría haber dado un resultado mejor variando esos valores.

Un aspecto que me costó resolver fue las casillas con idéntica puntuación. En ese caso, se le da prioridad a avanzar siempre, por lo que si hay casillas alcanzables delante no se plantea ir a los lados. Además, entre las que están delante, se elige la que está más cerca de una casilla desconocida (marcada con ? en MapaResultado) yendo en la dirección con la que iría a dicha casilla. Así se le da prioridad a descubrir nuevas zonas.

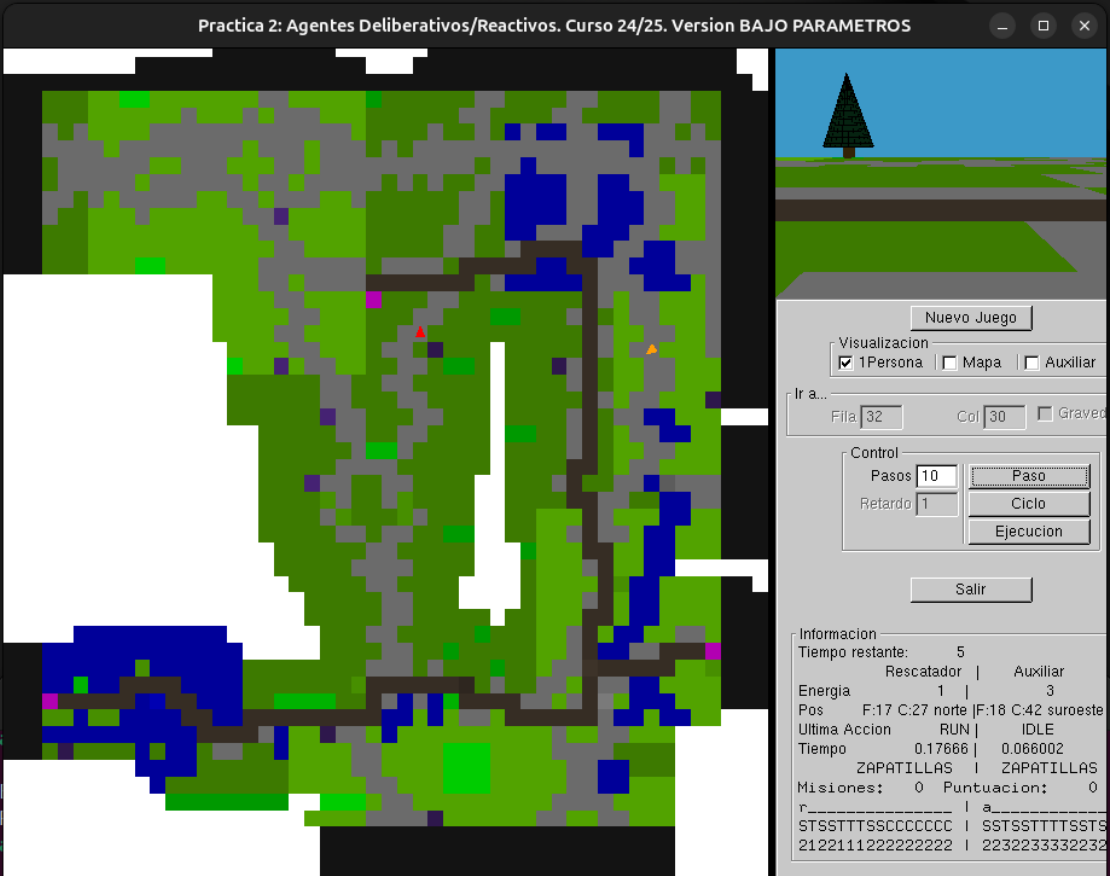
Por último, hice uso de las recargas. Tras descubrir la acción que más coste podía tener para él, fijé que cuando la energía restante fuese esa hiciese un IDLE para recargar energía. Para que la simulación no terminase por culpa del auxiliar, este también hace IDLE cuando va a morir, aunque este no recargue energía. Como podemos ver en las imágenes adjuntas, la gestión de las recargas es exacta puesto que siempre terminan los dos agentes prácticamente sin energía y la simulación se termina porque se agotan los instantes de simulación. Este es el objetivo buscado.

Algunas mejoras de este planteamiento son:

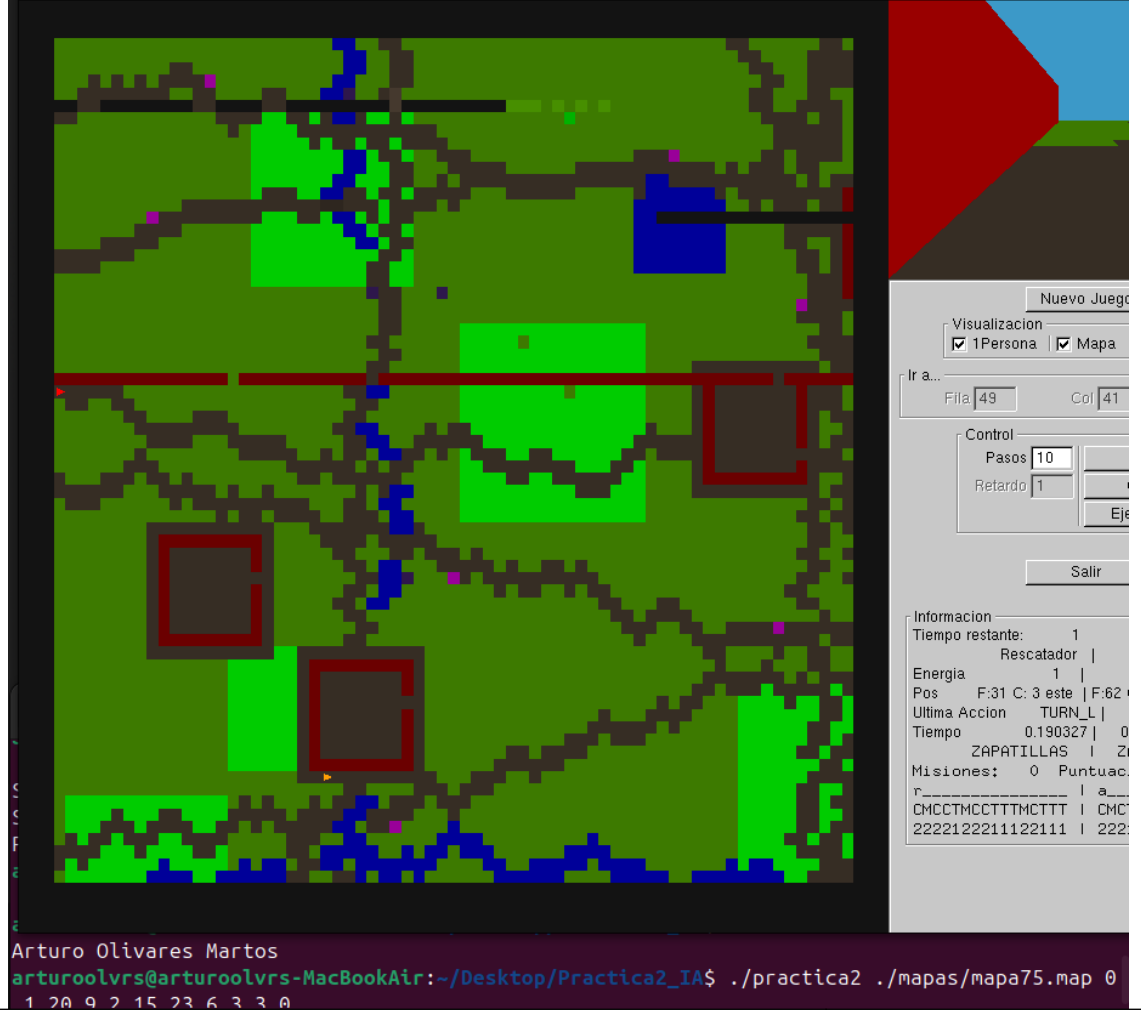
- Permitir, bajo condiciones muy limitadas, emplear otro tipo de casillas como matorral para descubrir más zonas del mapa. Esta implementación no es sencilla puesto que hay que hacer muchas pruebas para que este sobrecoste de energía no fuerce a que termine antes la simulación. Por tanto, no ha dado tiempo a implementarlo.
- Hacer más pruebas para determinar los valores más óptimos de PESO_VISTA y PESO_VISITADA

(b) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa50.map 0 1 5 3 2 36 44 6 3 3 0

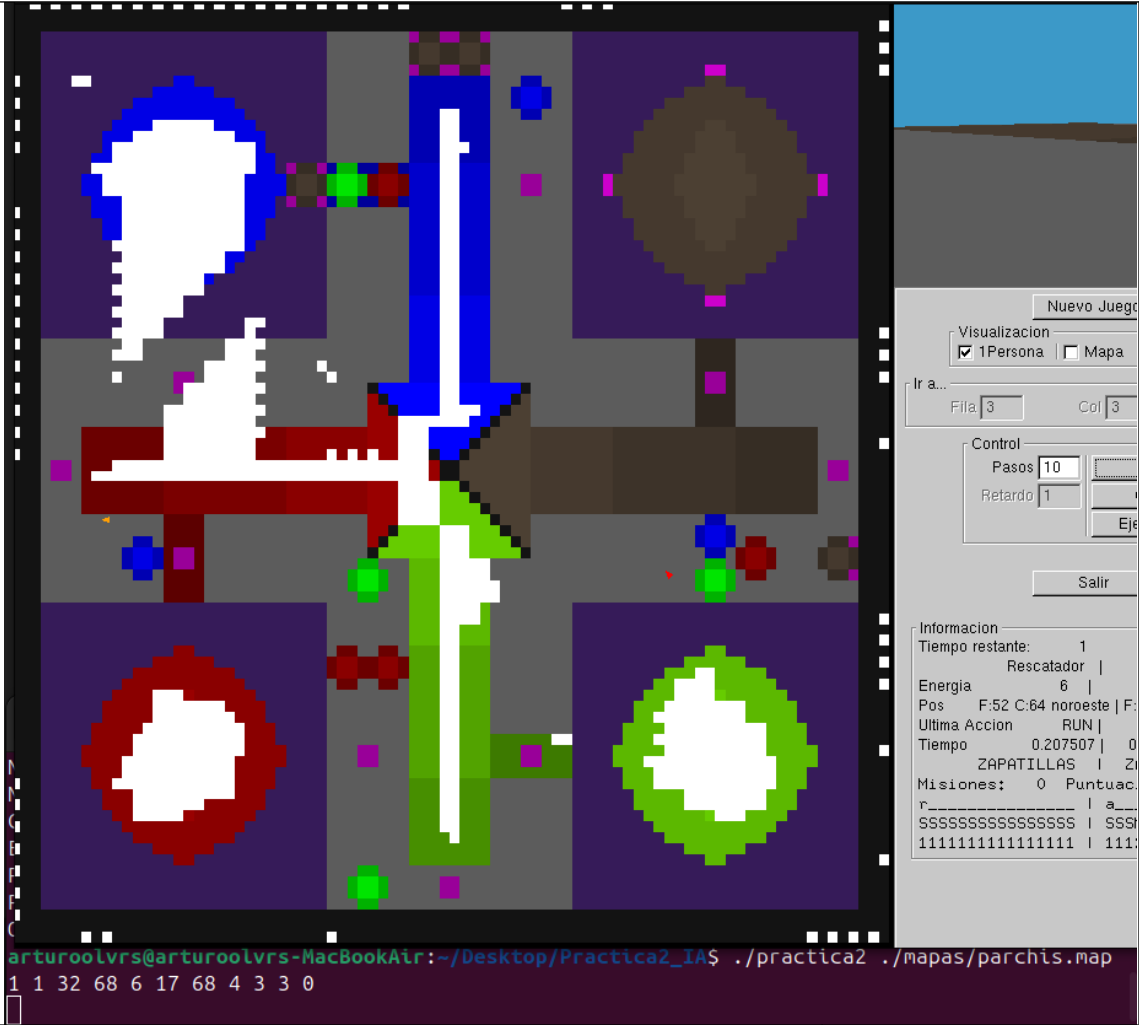
<p>Screen Shot</p>	 <pre> arturoolvrs@arturoolvrs-MacBookAir:~/Desktop/Practica2_IA\$ echo "Arturo Olivares Martos" Arturo Olivares Martos arturoolvrs@arturoolvrs-MacBookAir:~/Desktop/Practica2_IA\$./practica2 ./mapas/mapa50.map 0 1 5 3 2 36 44 6 3 3 0 </pre>
<p>Porcentaje descubierto de caminos y senderos</p>	<p>100</p>

(c) Rellena los datos de la tabla con el resultado de aplicar
`./practica2SG ./mapas/mapa75.map 0 1 20 9 2 15 23 6 3 3 0`

<p>Screen Shot</p>	 <p>The screenshot shows a game interface with a map on the left and a control panel on the right. The map is a grid with green and brown tiles, and a red path is visible. The control panel includes buttons for 'Nuevo Juego', 'Visualizacion', '1 Persona', 'Mapa', 'Fila', 'Col', 'Control', 'Pasos', 'Retardo', 'Salir', and 'Informacion'. The 'Informacion' panel displays game statistics such as 'Tiempo restante', 'Rescatador', 'Energia', 'Pos', 'Ultima Accion', 'Tiempo', 'Misiones', and 'Puntuacion'. At the bottom of the screenshot, a terminal window shows the command: <code>arturooolvrs@arturooolvrs-MacBookAir:~/Desktop/Practica2_IA\$./practica2 ./mapas/mapa75.map 0</code></p>
<p>Porcentaje descubierto de caminos y senderos</p>	<p>95.6752</p>

(d) Rellena los datos de la tabla con el resultado de aplicar
`./practica2SG ./mapas/parchis.map 1 1 32 68 6 17 68 4 3 3 0`

Screen
Shot



Porcentaje descubierto de caminos y senderos

97.4856

Nivel 2-La Búsqueda del Camino de Dijkstra

- (a) Indica cuál ha sido la definición de estado para resolver este problema. Justifica la definición.

Un estado se ha definido como un *struct* que contiene la posición del agente, la orientación, y si tiene o no zapatillas. Se ha empleado esta definición porque es la necesaria para determinar si el agente puede realizar cada acción.

- (b) ¿Has incluido dentro del algoritmo de búsqueda que si pasas por una casilla que da las zapatillas, considere en todos los estados descendientes de él, se está en posesión de las zapatillas? En caso afirmativo, explicar brevemente cómo.

Sí. Cuando voy a explorar un nodo que haya extraído de la cola frontera, antes de explorarlo considero si en esa casilla hay zapatillas y no las tiene. Si es así, el agente “coge” las zapatillas, es decir, actualiza ese estado. A partir de ahí, todos los descendientes se crean inicialmente copiando su antecesor, por lo que todos sus descendientes tendrán zapatillas. La implementación en código es la que sigue:

```
// Actualizamos el estado de las zapatillas
if (!nodoActual.estado.tieneZapatillas &&
    (mapaResultado.at(nodoActual.estado.fil).at(nodoActual.estado.col) == 'D'
     || mapaResultado.at(nodoActual.estado.fil).at(nodoActual.estado.col) == '?')
)
    nodoActual.estado.tieneZapatillas = true;
```

Notemos que, en este nivel, nunca se va a dar el caso de que la casilla sea '?'. Esto se plantea para el nivel 4.

- (c) En el algoritmo de búsqueda en anchura, el primer nodo que se genera compatible con la solución es la solución óptima y se puede detener la búsqueda en ese punto. Esto no se verifica en el algoritmo de Dijkstra. ¿Tuviste en cuenta esto en tu implementación? Describe brevemente cómo lo tuviste en cuenta.

Cuando un nodo se genera, tan solo se añade a la cola frontera, que se trata de una cola con prioridad ordenada según la energía empleada. El hecho de que se haya llegado a estado compatible con la solución no se comprueba ahí, sino al extraerlo de la frontera. Esto nos garantiza que cuando se extrae una solución de la frontera es el camino óptimo para llegar a esa casilla.

- (d) Incluye en el siguiente recuadro de texto el trozo de código de tu implementación del algoritmo de Dijkstra que genera el nodo descendiente de aplicar la acción RUN sobre el estado actual. Si tu proceso de generación de descendientes es genérico, pon como trozo de código todo el ciclo donde esté incluida esa generación de los descendientes.

El siguiente código es el genérico, en el que se genera el descendiente resultante de cada acción.

```
// Exploramos los nodos resultantes de aplicar cada una de las acciones
Nodo nuevoNodo;
bool ejecutada; // Determina si la acción ha podido ser ejecutada o no

for (auto it = acciones.begin(); it != acciones.end(); ++it){

    tie(nuevoNodo.estado, ejecutada) = ejecutarAccion(*it, nodoActual.estado, aEvitar);

    // Comprobamos que se ha generado un nodo nuevo.
    // Es necesario comprobar ambas, porque puede ser que se haya ejecutado pero haya llegado a un estado
ya visitado
    if (ejecutada && visitados.find(nuevoNodo) == visitados.end()){

        // Modificamos el gasto de energía del nuevo nodo
        nuevoNodo.gastoEnergia = nodoActual.gastoEnergia +
gastoAccion(*it, nodoActual.estado.fil, nodoActual.estado.col, nuevoNodo.estado.fil, nuevoNodo.estado.col);
        nuevoNodo.acciones = nodoActual.acciones;
        nuevoNodo.acciones.push_back(*it);

        // Añadimos el nuevo nodo a la frontera (podrá estar ya, pero podemos haber encontrado un camino
mejor)
        frontera.push(nuevoNodo);
    }
} // for accion
```

El siguiente código es el de la creación de cada descendiente. Para la creación del estado correspondiente a la acción de correr, vemos que tan solo se comprueba que la casilla 2 sea accesible (sin tener en cuenta la altura) y la casilla 6 esté también accesible (teniendo ahí en cuenta la altura). En este nivel, el parámetro *aEvitar* no se emplea (es siempre un set vacío). Se usará en el nivel 4, pero la función es la misma.

```
pair<ComportamientoRescatador::Estado, bool> ComportamientoRescatador::ejecutarAccion(Action action, const
Estado & inicio, const set<pair<int, int>>& aEvitar){

    Estado nuevoEstado = inicio;

    switch (action){
        case Action::RUN:
            if (casillaAccesible(inicio, false, 2, aEvitar) && casillaAccesible(inicio, true, 6, aEvitar)){
                tie(nuevoEstado.fil, nuevoEstado.col) = aCoordenadas(inicio.fil, inicio.col,
inicio.orientacion, 6);
            }
            break;
        case Action::TURN_SR:
            nuevoEstado.orientacion = (Orientacion)((((int)inicio.orientacion + 1) % 8);
            break;
        case Action::TURN_L:
            nuevoEstado.orientacion = (Orientacion)((8+(int)inicio.orientacion - 2) % 8);
            break;
        case Action::WALK:
            if (casillaAccesible(inicio, true, 2, aEvitar)){
                tie(nuevoEstado.fil, nuevoEstado.col) = aCoordenadas(inicio.fil, inicio.col,
inicio.orientacion, 2);
            }
            break;
    }
}
```

```

    }

    break;
}

return make_pair(nuevoEstado, !(nuevoEstado == inicio));
}

```

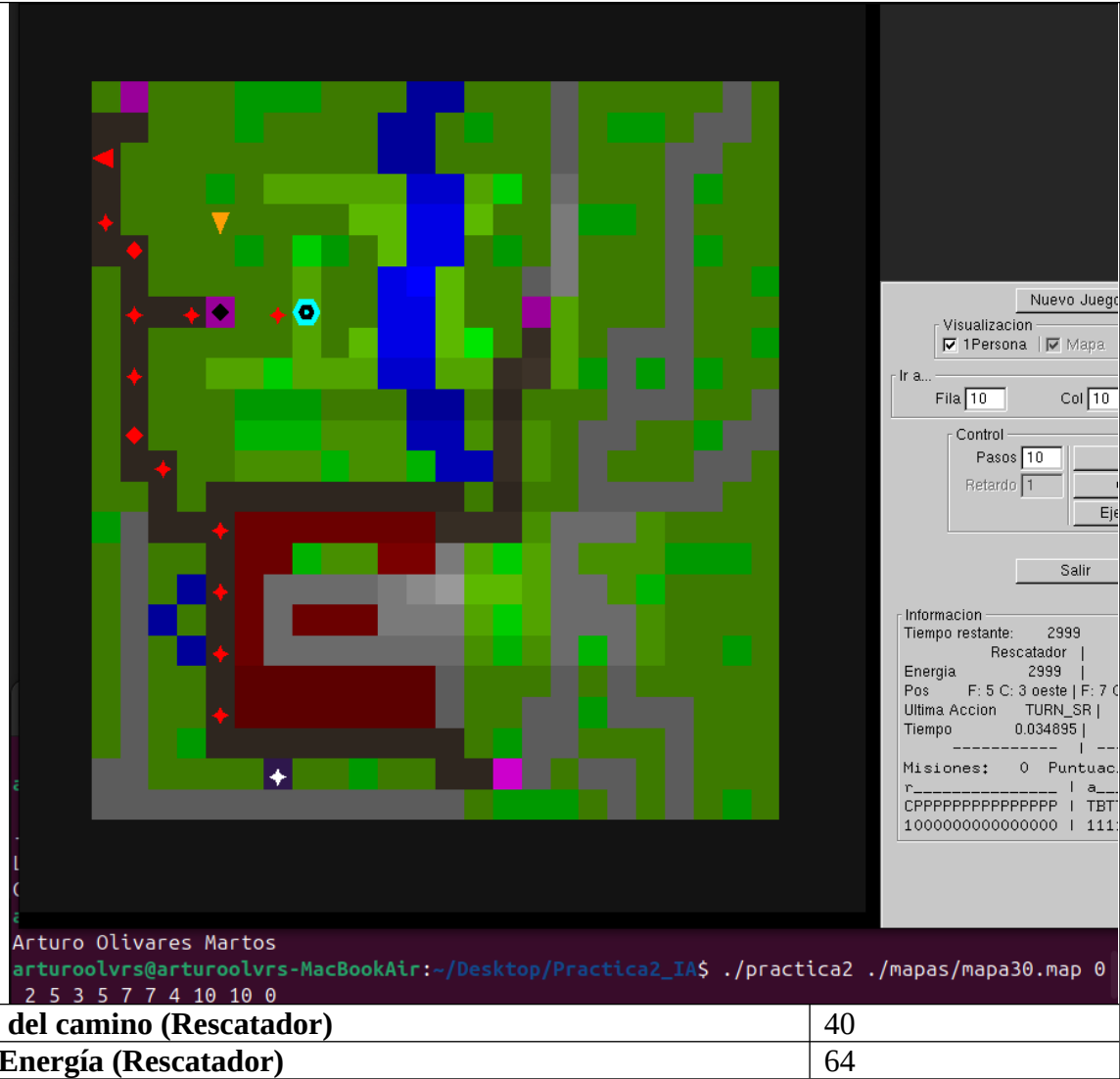
(e) Rellena los datos de la tabla con el resultado de aplicar
./practica2SG ./mapas/mapa30.map 1 2 11 4 4 26 26 0 3 4 0

Screen
Shot

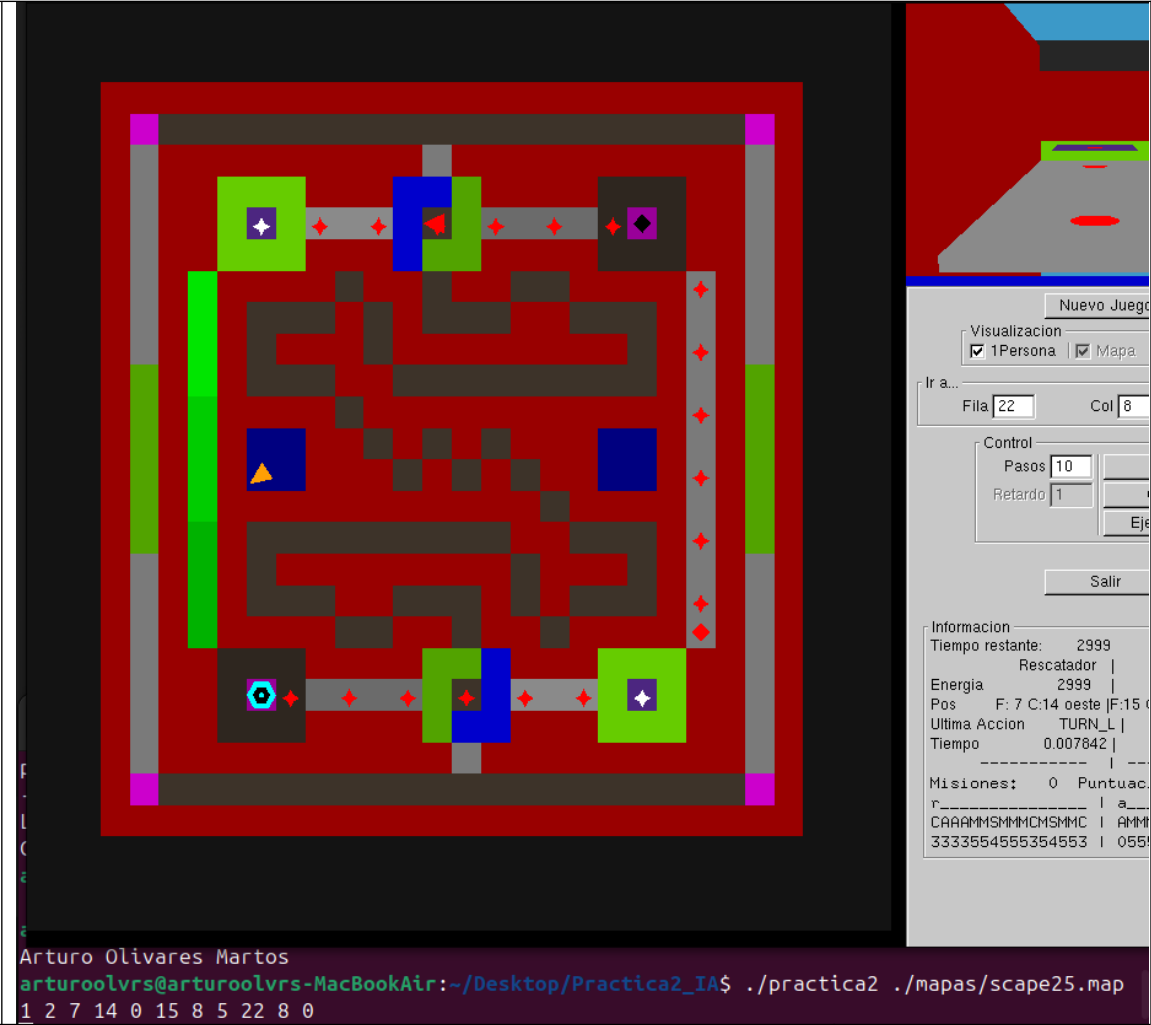


Longitud del camino (Rescatador)	11
Coste de Energía (Rescatador)	11

(f) Rellena los datos de la tabla con el resultado de aplicar
`./practica2SG ./mapas/mapa30.map 0 2 5 3 5 7 7 4 10 10 0`

<p>Screen Shot</p>	 <p>Longitud del camino (Rescatador) 40</p> <p>Coste de Energía (Rescatador) 64</p>
--------------------	--

(g) Rellena los datos de la tabla con el resultado de aplicar
./practica2SG ./mapas/scape25.map 1 2 7 14 0 15 8 5 22 8 0

<div>Screen Shot</div>	
Longitud del camino (Rescatador)	32
Coste de Energía (Rescatador)	61

Nivel 3-El Ascenso del A*uxiliar

- (a) ¿Qué diferencia este algoritmo del de Dijkstra 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)

Además de los cambios evidentes (como en la función *CasillaAccesible*) o permitir acciones diferentes a las del rescatador, la única diferencia está en la ordenación de la cola con prioridad frontera. Ahora, en vez de ordenarse tan solo por la energía empleada en llegar hasta dicho estado, se ordena en función de ese valor sumado al valor de la heurística del coste estimado en llegar desde el dicho estado hasta el destino. La heurística se explica en la siguiente pregunta.

Estas diferencias se muestran a continuación:

```
// Cola con prioridad formada por Nodos. Ordenada por el gasto de Energia al nodo Origen
struct Comparador {
    int filDestino;
    int colDestino;

    Comparador(int fil, int col) : filDestino(fil), colDestino(col) {}

    bool operator()(const Nodo& a, const Nodo& b) const {
        return a.gastoEnergia + Heuristica(a.estado, filDestino, colDestino)
            > b.gastoEnergia + Heuristica(b.estado, filDestino, colDestino);
    }
};

priority_queue<Nodo, vector<Nodo>, Comparador> frontera(Comparador(destReferencia.first,
destReferencia.second));
```

- (b) Copia y pega en el siguiente recuadro de texto la heurística seleccionada. Además, descríbela y justifica la razón que hace que sea admisible para este problema.

```
int ComportamientoAuxiliar::Heuristica(const Estado& estado, int filDestino, int colDestino)
{
    // Heurística de la Norma Infinito
    int difFil = abs(estado.fil - filDestino);
    int difCol = abs(estado.col - colDestino);
    return max(difFil, difCol);
}
```

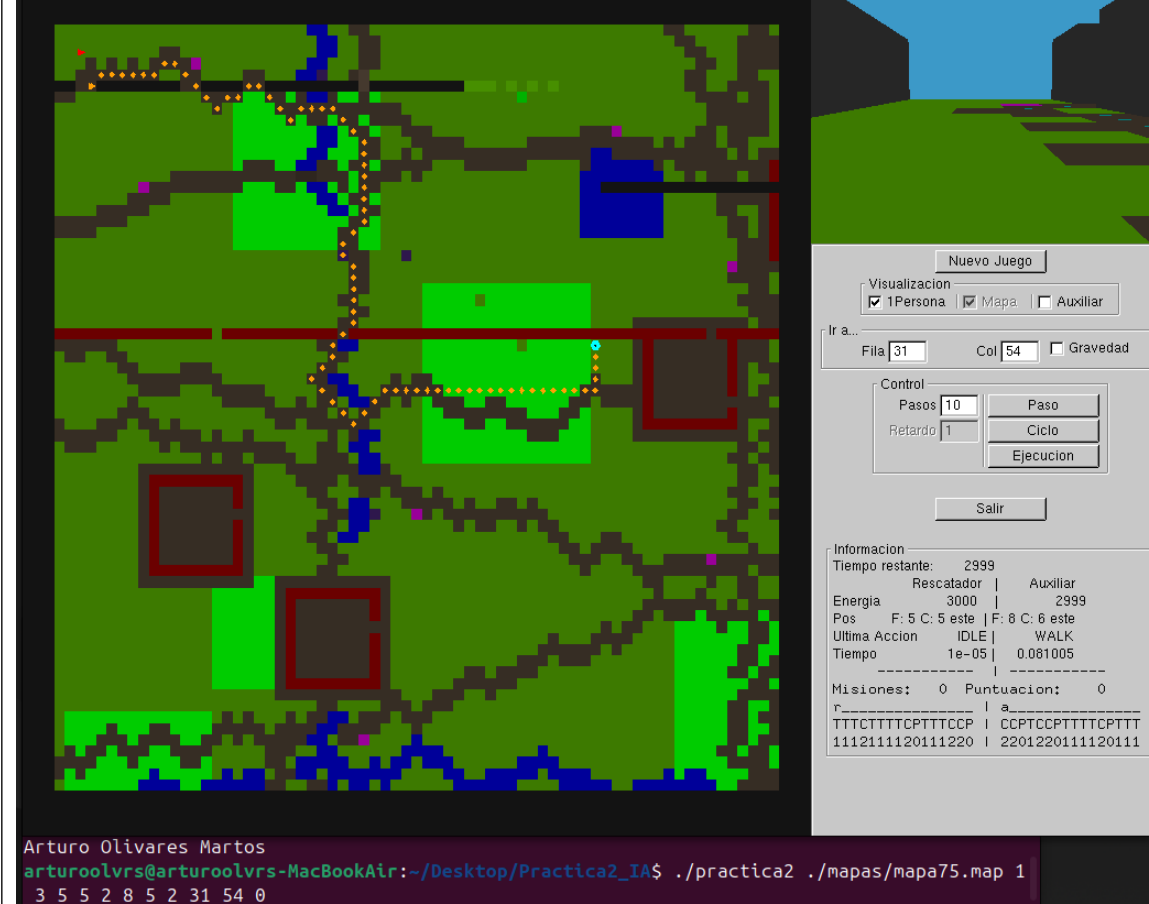
Esta heurística es conocida como la distancia infinito o la distancia del máximo. Es el máximo entre el valor absoluto de la diferencia de filas y la diferencia de columnas. Veamos por qué es válida.

- No sobreestima el coste real de alcanzar el nodo objetivo. Por tanto, es admisible. Esto se debe a que todas las acciones tienen coste mayor que uno y el auxiliar no puede correr, por lo que siempre ha de recorrer todas las casillas. De hecho, si el auxiliar no pudiese andar en diagonal podríamos emplear la distancia de Manhattan, pero como sí puede hemos de emplear la norma infinito, puesto que está garantizado que, como


- mínimo, visita el número de casillas dado por la heurística.
- Además, es válida y permite no hacer más cambios en el código porque, si $h(x)$ es el valor de la heurística para llegar al objetivo desde el nodo x y $g(x)$ es el valor del coste real en alcanzar el nodo x desde el origen, entonces se cumple que
$$h(x) \leq g(y) - g(x) + h(y)$$

(c) Rellena los datos de la tabla con el resultado de aplicar


`./practica2SG ./mapas/mapa75.map 1 3 5 5 2 8 5 2 31 54 0`

<p>Screen Shot</p>	
<p>Longitud del camino (Auxiliar)</p>	<p>165</p>
<p>Coste de Energía (Auxiliar)</p>	<p>203</p>

(d) Rellena los datos de la tabla con el resultado de aplicar
./practica2SG ./mapas/2ez.map 0 3 7 7 4 14 16 4 16 16 0

<div>Screen Shot</div>	
Longitud del camino (Auxiliar)	157
Coste de Energía (Auxiliar)	365

(e) Rellena los datos de la tabla con el resultado de aplicar
`./practica2SG ./mapas/paldea25.map 0 3 82 17 4 16 23 6 34 34 0`

<div data-bbox="172 645 268 712">Screen Shot</div>	 <p>The screenshot shows a game interface. On the left is a map with a green and blue terrain, a red path, and a black circle. On the right is a control panel with buttons for 'Nuevo Juego', 'Visualizacion' (1 Persona, Mapa, Auxiliar), 'Fila' (34), 'Col' (34), 'Gravedad', 'Control' (Pasos: 10, Retardo: 1, Paso, Ciclo, Ejecucion), and 'Salir'. Below the map is a terminal window showing the command: <code>arturoolvr@arturoolvr-MacBookAir:~/Desktop/Practica2_IA\$./practica2 ./mapas/paldea25.map 0 3 82 17 4 16 23 6 34 34 0</code></p>
Longitud del camino (Auxiliar)	51
Coste de Energía (Auxiliar)	246

Nivel 4-Misión de Rescate

- (a) Haz una descripción general de tu estrategia general con la que has abordado este nivel. Explica brevemente las razones de esos criterios.

Este nivel, habiendo ya programado los niveles anteriores, ha sido programado de forma natural y como se cabe esperar. Los aspectos que difieren especialmente en este nivel se encuentran detallados en las siguientes preguntas.

Como cabría esperar, el agente rescatador va siempre a la casilla objetivo. Inicialmente, cuando el mapa no se conoce, es necesario controlar:

- Qué hacer con las casillas desconocidas. Pregunta d)
- Cómo controlar el tema de las zapatillas. Pregunta f)

También, como el mapa no es estático sino dinámico, el plan realizado no siempre puede llevarse a cabo. Es por ello necesario controlar qué hacer cuando un plan no puede llevarse a cabo y cuándo es necesario replanificar. Esto se desarrolla en la pregunta c).

Una vez que el agente llega al destino, si el herido no es de gravedad directamente pasa al siguiente objetivo y se reitera el proceso ya descrito. Si es de gravedad, hace siempre un CALL_ON (esto ha sido añadido como un aspecto a mejorar, desarrollado en f), puesto que no tiene por qué ser la decisión óptima).

Cuando se hace el CALL_ON, el auxiliar va a una casilla desde la que se ve el destino (no tiene por qué ser el destino. Esto se desarrolla en b)). Una vez llegue al destino, la misión se dará por terminada y se volverá a iniciar el proceso.

Entre todo este proceso, es necesario gestionar la recarga de energía de los agentes. Este aspecto se desarrolla en la pregunta e).

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

Uso los mismos algoritmos de búsqueda ya implementados en los niveles 2 y 3, con ciertas variantes puesto que ahora el mapa inicialmente es desconocido y, además, es dinámico.

- Para el rescatador, uso el algoritmo de Dijkstra.
- Para el auxiliar, uso el algoritmo A* con la heurística previamente mencionada.

Me planteé modificar el algoritmo del rescatador, pero no fue necesario porque no me daba problemas ningunos tal y como estaba programado e implementado. No tenía problemas ninguno con el límite de tiempo, por lo que opté por usar esos mismos algoritmos.

La principal modificación que le he hecho a ambos algoritmos es que no reciben como destino una única casilla, sino un conjunto de casillas. Esto tiene las siguientes consecuencias:

- A la hora de recargar, directamente se le pasa como parámetro el conjunto de todos los puestos base que ese agente ha detectado en el mapa, y así se obtiene el plan al puesto base más cercano.
- El auxiliar no recibe como destino tan solo la casilla objetivo, sino todas las casillas desde la que se ve la casilla destino. Esto permite que, si no puede alcanzar el destino pero sí puede verlo, proporcione un plan.

(c) ¿Bajo qué condiciones replanifica tu agente?

Puesto que el algoritmo A* es más rápido y no tenía problemas de tiempo, el auxiliar replanifica en cada iteración. Aunque esto sea más costoso computacionalmente, esto le permite decidir siempre la acción más óptima según esté el mapa y según lo que conozca. Además, así se evita que se choque con los agentes presentes en el mapa dinámico.

El rescatador sí tuvo que modificarlo, puesto que tardaba demasiado (era costoso comprobar y visualizarlo). Replanifica en las siguientes circunstancias:

- En la iteración actual ha descubierto casillas nuevas que antes no tenía. Esto permite que cada vez que descubra nuevas casillas replanifique obteniendo así un camino cada vez más óptimo.
- No tiene plan a ejecutar. Esto permite que planifique cuando le cambian el destino.
- Ha sido empujado, y por tanto se activa el sensor de choque. Si ha sido empujado, posiblemente no se encuentre en la casilla que tenía marcada en su plan inicial, por lo que posiblemente no sea posible seguir su plan inicial y ha de replanificar.
- El agente detecta que no puede ejecutar la acción. Es importante tener en cuenta que si no puede ejecutarla no la ejecuta (nunca llega a producirse el choque). Esto permite que no se pierdan instantes de simulación, sino que directamente si detecta que no puede ejecutar la que tiene en el plan, entonces replanifica y ejecuta una que sí pueda. Esto se puede deber a que tenga excursionistas o vándalos delante, por ejemplo.
- Por último, también replanifica si antes no tenía zapatillas y, en esa iteración, ha conseguido zapatillas. Esto se debe a que, así, la planificación va a ser más certera y correcta. Este aspecto lo desarrollo de forma más extensa en la pregunta f).

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

El coste que les he asignado es el coste por defecto de las casillas, es decir, el coste más bajo que hay (el mismo coste que para caminos, por ejemplo). Esto es así para fomentar que el agente inicialmente avance hasta este tipo de casillas, avanzando así en el descubrimiento del mapa.

(e) ¿Has tenido en cuenta la recarga de energía? En caso afirmativo, describe la política usada por los agentes.

El primer agente que me planteé que necesitaba recargar fue el rescatador, puesto que siempre se quedaba sin energía. Para esto, el agente va almacenando en un set los puestos base que ve durante sus recorridos, aunque no vaya a ellos. Esto le permite que, una vez se quiere recargar, se elija de esos puestos base el más cercano empleando el algoritmo de Dijkstra como he mencionado en la pregunta b).

Esta idea tiene el siguiente problema:

- Si los objetivos de un mapa están muy concentrados en una zona y el agente nunca descubre un puesto base, cuando necesite recargar no va buscar en zonas desconocidas un puesto base, por lo que directamente no recargará. No obstante, esta situación es muy específica, no sucedía en ningún test, y no era de fácil solución, por lo que directamente no se considera.

La gestión de cuándo recargar es lo difícil y complicado. Contemplo dos situaciones.

- El rescatador está prácticamente sin energía y va a morir dentro de poco. Este límite lo mido con la constante `VIDA_A_RECARGAR`, que he fijado a 200. Cuando al agente le

quedan menos de 200 de energía y quedan más de 200 instantes de simulación, entonces se va a recargar. Este valor debe ser suficientemente alto para que le dé tiempo a llegar al puesto base. Tan solo se recarga si quedan suficientes instantes de simulación, puesto que no tiene sentido recargar si la simulación se va a terminar.

- Cuando el rescatador hace CALL_ON, el rescatador está una serie de instantes de simulación en la casilla destino sin hacer nada, por lo que se desperdician instantes de simulación. Planteé por tanto siempre que hace CALL_ON ir a recargar, aprovechando así esos instantes de simulación. La cantidad de energía que se recarga la he fijado dependiendo del tamaño del mapa, puesto que si el mapa es más grande el auxiliar tardará más en llegar.

Cuánta energía recargar en cada instante, con cuántos instantes ir a recargar, etc. son decisiones complejas que no he podido terminar de refinar y llegar a los valores óptimos.

Respecto al auxiliar, puesto que comprobé que en ninguno de los test se terminaba porque el auxiliar se quedase sin energía y, además, la energía final del auxiliar siempre era alta, opté por no implementar este aspecto.

- (f) Añade aquí todos los comentarios que desees sobre el trabajo que has desarrollado sobre este nivel, qué consideras 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.

En primer lugar, he de destacar cómo controlo el tema de las zapatillas. Los últimos tests me daban error, puesto que el agente no tenía zapatillas y al planificar tampoco las cogía (puesto que la casilla en la que estaban las zapatillas estaba oculta). Por tanto, no tenía camino posible para alcanzar el destino.

Esto conseguí evitarlo estableciendo en mis algoritmos de búsqueda que, cuando se explora un estado cuya casilla es desconocida, se asuma en la planificación que el agente coge las zapatillas ahí. Esto nos permite que, mientras que haya casillas desconocidas en el plan, el agente suponga que ahí puede haber zapatillas y así, si las necesita, pase por ahí. Este es también el motivo por el cual se replanifica una vez obtenga realmente las zapatillas, porque así las planificaciones serán más certeras al no tener que hacer esa suposición.

También destacar que el rescatador, cuando llega y detecta que la gravedad del herido es alta, siempre hace CALL_ON, puesto que el número de puntos dado si llega el auxiliar es bastante mayor.

Otro riesgo que merece la pena comentar es que, cuando el auxiliar llegaba a la casilla desde la cual se ve la posición de destino, no tiene por qué tener la orientación correcta. Por ello, una vez que llega, realiza TURN_SR hasta que vea a través del sensor la casilla objetivo.

Algunas mejoras que podría añadirle a mis agentes, de haber tenido tiempo, sería:

- En el caso de que sea necesario recargar y el agente no haya descubierto algunas zonas, sería buena idea plantear que el agente vaya a descubrir dichas zonas para así buscar puestos base. No se ha implementado porque no se daba esta situación tan concreta en ninguno de los mapas, y es una decisión arriesgada puesto que puede ser que en el intento de búsqueda no llegue a nada.
- Implementar la recarga para el auxiliar, por si en algún caso concreto fuese necesario. Podría recargar mientras que no tiene ningún destino válido, y en el momento que lo

reciba dejar de recargar e ir directamente.

- Realizar una mayor cantidad de test, pruebas, etc. con distintos valores de los parámetros de la recarga para que el auxiliar nunca tenga que quedarse esperando a que llegue el rescatador de recargar, y que la recarga sea por tanto más eficiente.
- Hacer más pruebas y plantear distintos comportamientos de los agentes. Se podría haber planteado que, cuando el auxiliar no tenga destino válido, se vaya directamente al centro del mapa para así tardar menos. También se podría haber planteado que, en ciertas condiciones, el Rescatador haga CALL_OFF en vez de CALL_ON. Por último, también se podría contemplar que el rescatador haga siempre CALL_ON, hasta antes de llegar al destino, para que así el auxiliar vaya yendo y, en el caso de que sea urgente, tarde menos en llegar. Todas estas pruebas son arriesgadas, y habría que contemplar si mejoran o no la solución.

(g) Rellena los datos de la tabla con el resultado de aplicar

```
./practica2SG ./mapas/mapa50.map 1 4 28 25 4 28 20 2 36 23 0 39 8 0 46 26 1 39 34 0 26 37 0
18 46 0 3 46 0 3 3 0 10 17 1 39 45 0 9 16 0 38 13 0 27 23 0 31 18 0 45 31 0 35 7 0 12 6 1 40 7 0 20 6
1 10 25 1 41 30 0 14 31 0 26 24 1 38 26 1 38 20 1 44 14 0 17 40 0 45 3 1 4 9 0 33 44 0 17 3 1 3 11 0
42 13 1 26 18 1 38 25 1 33 26 0 46 46 1 36 14 0 36 31 1 17 34 0 8 22 1 44 41 1 16 11 0 44 17 0 29
32 0 42 21 0 46 19 1 40 34 0 45 24 0 46 7 0 44 32 1 21 30 1 14 39 1 15 22 1 11 9 0 13 27 1 20 8 1 45
5 0
```

Instantes de simulación no consumidos	0
Tiempo Consumido	7.19052
Nivel Final de Energía (Rescatador)	1969
Nivel Final de Energía (Auxiliar)	212
Objetivos encontrados	(40) 155

(h) Rellena los datos de la tabla con el resultado de aplicar

```
./practica2SG ./mapas/mapa100.map 1 4 63 31 6 63 32 2 66 40 0 75 24 0 85 36 1 83 6 0 60 10 0
33 11 0 84 7 0 86 40 0 68 77 1 79 91 0 19 33 0 76 25 0 55 47 0 62 36 0 51 95 0 91 63 0 71 14 1 24
13 0 80 15 1 21 51 1 83 61 0 29 63 0 52 49 1 78 52 1 76 40 1 90 28 0 39 80 0 91 6 1 94 52 0 8 19 0
66 89 1 34 6 0 6 23 1 85 26 1 53 37 1 79 51 0 70 53 1 3 43 0
```

Instantes de simulación no consumidos	0
Tiempo Consumido	2.9288
Nivel Final de Energía (Rescatador)	1044
Nivel Final de Energía (Auxiliar)	1797
Objetivos encontrados	(26) 92

(i) Rellena los datos de la tabla con el resultado de aplicar

```
./practica2SG ./mapas/F_islas.map 1 4 47 53 2 49 53 2 41 56 0 52 53 0 74 54 1 74 47 0 46 42 0 71
56 0 83 52 0 58 65 0 85 43 1 92 39 0 81 68 0 91 48 0 21 95 0 92 14 0 88 64 0 43 61 0 28 78 1 30 44
0 22 18 1 27 55 1 41 16 0 90 10 0 12 49 1 76 68 1 38 74 1
```

Instantes de simulación no consumidos	0
Tiempo Consumido	6.78408
Nivel Final de Energía (Rescatador)	1808
Nivel Final de Energía (Auxiliar)	1290
Objetivos encontrados	(37) 159

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.

De cara a futuros años, recomiendo muy encarecidamente que, de alguna forma, podamos también modificar la clase Comportamiento para hacer uso de la herencia. Hay una gran cantidad de funciones del auxiliar que son copia exacta de las del rescatador, y sino muy parecidas. Esto habría tenido las siguientes ventajas:

- Habría ahorrado una gran cantidad de horas de depuración, puesto que había veces que hacías cambios en uno de los agentes y se te olvidaba hacer el cambio análogo en el otro agente.
- Permitiría haber implementado muchos aspectos en el auxiliar que no hemos podido por cuestión de tiempo, pero que sí hemos implementado en el resaltador. De existir la herencia, al implementarlo para uno de los agentes estaríamos implementándolo prácticamente de forma automática también para el otro.

Por otro lado, el bot ha sido muy cómodo de cara a hacer las pruebas y la leaderboard es muy útil de cara a mejorar la práctica.