Proyecto de Simulación
Agentes

Dayany Alfaro González-C411

1. Principales ideas seguidas para la solución del problema

Para darle solución al problema se hizo uso del lengaje Python como herramienta. Se modeló el ambiente como una clase con ciertas funcionalidades y los elementos que interactúan en este como una jerarquía de clases.

Al inicio de una simulación se garantiza que el ambiente construido es factible. Primero se colocan los corrales de los niños, luego los niños garantizando que ninguno se encuentre inicialmente en el corral. Después se coloca la suciedad y, por último, los obstáculos teniendo en cuenta que estos no bloqueen el camino del robot hacia ningún elemento del ambiente. El robot va a existir independientemente del ambiente, este se garantiza que inicia en una casilla vacía y sin cargar a ningún niño.

Se proponen dos modelos de agentes para describir el comportamiento del robot. La estrategia de ambos modelos puede ser considerada híbrida entre reactivo y proactivo, pero cada uno presenta matices más marcados de alguno de los rasgos.

El desempeño de ambos modelos fue evaluado en 10 configuraciones de ambientes iniciales diferentes y para cada una de ellas se ejecutaron 30 simulaciones.

2. Modelos de agentes considerados

2.1. Modelo 1

El primer modelo que se propone se puede clasificar como un híbrido entre proactivo y reactivo, donde destaca el comportamiento proactivo. La estrategia que sigue se basa en priorizar la ubicacioón de los niños en el corral por encima de limpiar la suciedad. Si la suciedad se ha acumulado demasiado entonces el robot priorizará la limpieza.

Para recoger un niño analiza la distancia a estos y selecciona el más cercano a su posición. Una vez ha recogido un niño selecciona el corral mas distante a su posición (esto para evitar que al colocar niños en el corral estos bloqueen la entrada a el robot a los corrales que estén detrás y sean inaccesibles desde otras posiciones) y se dedica exclusivamente a llevarlo a ese corral. En el proceso de moverse hacia un niño para recogerlo o guardarlo en el corral, si el robot se encuentra encima de una casilla sucia, este no la limpiará porque prioriza por encima de todo a los niños.

La mayor manifestación de reactividad está dada en que, si en algún turno detecta que la suciedad en el ambiente ha sobrepasado el 50 %, entonces pasa a priorizar la limpieza de suciedad hasta que esta vuelva a estar por debajo del 50 % y entonces vuelve a retomar la recogida de niños. Si en el momento en que detecta este aumento de suciedad está cargando un niño primero culminará su objetivo de llevarlo al corral y después se dedicará al control de la suciedad.

Para limpiar la suciedad el robot simplemente encuentra la casilla más cercana que esté sucia y se mueve hacia ella para limpiarla.

Este modelo va a estar encapsulado en la clase `ChildsFirstRobot`.

2.2. Modelo 2

El segundo modelo que se propone también se puede clasificar como un híbrido entre proactivo y reactivo, pero en este caso destaca el comportamiento reactivo. La estrategia de este robot va a estar basada en la distancia de su posición al resto de los elementos del ambiente.

En cada turno se compara la distancia hacia las suciedades y los niños en el ambiente. Si lo más cercano al robot es un niño entonces se va a dedicar a buscar ese niño y luego llevarlo al corral mas distante de su posición. En el caso de que lo más cercano sea una suciedad entonces se va a centrar en ir y limpiar esa casilla sucia.

Si en el trayecto del robot hacia el niño o la suciedad que está más cerca de su posición ocurre un cambio en el ambiente que hace que algún otro elemento esté más cercano entonces el robot reajustará el objetivo hacia el cual se mueve.

Este modelo va a estar encapsulado en la clase `NearFirstRobot`.

3. Ideas seguidas para la implementación

La modelación del ambiente se encuentra en el archivo **environment.py**. Se tiene una clase `Environment` que contiene los siguientes atributos y métodos que permiten describir un ambiente:

- `matrix`: es un diccionario que tiene como llave una tupla `(i,j)` y como valor el objeto que representa al elemento que se encuentra en la posición `(i,j)` del ambiente o `None` en caso de que esté vacía.
- `robot`: contiene el objeto `Robot` que actúa como agente en el ambiente.
- `dirty_count`: sirve para registrar los niveles de suciedad en cada turno y luego calcular el nivel de suciedad promedio.
- `set_playpen()`: coloca el corral en el ambiente.
- `initialize()`: coloca niños, suciedad y objetos en el ambiente.
- `initialize_robot()`: crea el robot y lo coloca en una posición vacía del ambiente.
- `generate_dirtiness()`: se encarga de seleccionar las casillas que van a ser ensuciadas según la ubicación de los niños en el ambiente.

- `natural_change()`: realiza las acciones que describen un cambio natural del ambiente.
- `random_change()`: realiza las acciones que describen un cambio aleatorio del ambiente.

Los elementos del ambiente se encuentran modelados en el archivo **elements.py**. El diseño consiste en una clase base `Element`. Esta clase está formada por los siguientes atributos y métodos:

- `pos`: es una tupla `(i,j)` que describe la posición del elemento.
- `environment`: es un objeto de tipo `Environment` que describe el ambiente al que pertenece el elemento.
- `find_next_step()`: dada una dirección a la que moverse calcula la posición en la que te colocarías.
- `step()`: cambia la posición del elemento en el ambiente.

`Element` va a ser la clase base para las siguientes:

- `Child`: representa un niño.
  Esta clase además va a definir los siguientes métodos:
  - `move`: realiza el movimiento de los niños en el ambiente.
- `Obstacle`: representa un obstáculo.
  Esta clase además va a definir los siguientes métodos:
  - `move`: realiza el movimiento de los obstáculos cuando son empujados por los niños.
- `Dirty`: representa una suciedad.
- `Playpen`: representa un corral.
  Esta clase además va a definir los siguientes atributos:
  - `child`: es un *bool* que indica si el corral contiene o no un niño.
- `Robot`: representa el robot.
  Esta clase va a definir los siguientes atributos y métodos:
  - `child`: es un *bool* que indica si el robot tiene cargado o no un niño.
  - `bfs()`: calcula la distancia y el camino hacia todos los elementos del ambiente.
  - `get_path()`: devuelve el camino hacia un elemento determinado del ambiente.
  - `find_near_element()`: dado un conjunto de elementos devuelve el más cercano de ellos.
  - `find_far_element()`: dado un conjunto de elementos devuelve el más lejano de ellos.

  Los modelos de agentes implementados `NearFirstRobot` y `ChildsFirstRobot` van a heredar de esta clase y definir sus estrategias en un método llamado `move()`.

En el archivo **main.py** es donde se definen los ambientes iniciales, se ejecutan las simulaciones y se guarda la información que describe los resultados.

4. Consideraciones obtenidas a partir de la ejecución de las simulaciones del problema

Se construyeron ambientes iniciales con las características que se muestran en el Cuadro 1.

Parámetros						
No. Ambiente	N	M	Niños	Porciento de Suciedad	Porciento de Obstáculos	t
1	10	10	5	30	20	10
2	7	8	3	20	10	5
3	7	8	3	20	10	20
4	15	15	10	20	20	50
5	5	5	2	10	5	5
6	10	5	3	30	20	10
7	10	10	5	40	10	20
8	10	10	5	10	40	30
9	9	9	3	30	20	20
10	10	10	4	20	20	20

Cuadro 1: Ambientes iniciales.

Cada uno de los modelos de agente implementados se colocó en ambientes con cada una de las características descritas en el Cuadro 1 y se realizaron 30 simulaciones en cada tipo de ambiente.En el Cuadro 2 se reportan los resultados obtenidos con el Modelo 1, mientras que en el Cuadro 3 se reporta el comportamiento del Modelo 2. Además esta información se encuentra en el archivo **output.txt**, hacia donde se guarda la salida de la ejecución de las simulaciones.

Resultados				
No.	Casa Limpia y Niños en Corral	Despedido	Tiempo Agotado	Media del Porciento de Casillas Sucias
1	28	0	2	23.32
2	28	0	2	14.6
3	24	0	6	12.67
4	30	0	0	17.91
5	22	0	8	5.9
6	25	0	5	23.06
7	29	0	1	26.84
8	28	0	2	11.37
9	28	0	2	21.78
10	26	0	4	14.84

Cuadro 2: Modelo 1.

Resultados				
No.	Casa Limpia y Niños en Corral	Despedido	Tiempo Agotado	Media del Porciento de Casillas Sucias
1	29	0	1	17.2
2	21	0	9	7.67
3	28	0	2	9.46
4	30	0	0	11.24
5	18	0	12	2.86
6	26	0	4	15.81
7	28	0	2	19.34
8	28	0	2	7.66
9	29	0	1	16
10	26	0	4	10.07

Cuadro 3: Modelo 2.

Al analizar los resultados obtenidos se observa que ambos modelos presentaron un buen desempeño, dado que nunca fueron despedidos y, como se observa en el Cuadro 1, los niveles de suciedad iniciales y la cantidad de niños en el ambiente por lo general son significativos. Es razonable que nunca sean despedidos dado que el modelo 1 reacciona cuando el porciento de suciedad se está elevando y se dedica a mantenerlo bajo control, mientras que el modelo 2 va a dedicar su atención al niño o suciedad más cercano a su posición y es más probable que tenga cerca a una suciedad que un niño pues estas los superan en número generalmente. Por tanto en caso de que la suciedad sea demasiada ambos modelos simplemente se les va a acabar el tiempo en la tarea de limpieza de suciedad pero siempre la van a mantener bajo control. Además se puede apreciar que la media del porciento de suciedad siempre es mayor en el primer modelo, lo cual tiene sentido dado que ese modelo otorga mayor prioridad a colocar los niños en el corral, mientras que el segundo va realizando ambas tareas a la vez.