Informe del Proyecto Agentes. Simulación. Curso 2020-2021

Leonel Alejandro García López L.GARCIA3@ESTUDIANTES.MATCOM.UH.CU
Grupo C412

Tutor(es):
Dr. Yudivián Almeida Cruz, *Facultad de Matemática y Computación, Universidad de La Habana*
Lic. Gabriela Rodriguez Santa Cruz Pacheco, *Facultad de Matemática y Computación, Universidad de La Habana*
Lic. Daniel Alejandro Valdés Pérez, *Facultad de Matemática y Computación, Universidad de La Habana*

Tema: Simulación, Eventos Discretos

1. Introducción

Como proyecto se propuso simular un ambiente donde intervienen agentes correspondiente a una casa. El ambiente es discreto y tiene la forma de un rectángulo N x M. El ambiente es de información completa, por tanto todos los agentes conocen toda la información sobre el agente. El ambiente puede variar aleatoriamente cada t unidades de tiempo. El valor de t es conocido. Las acciones que realizan los agentes ocurren por turnos. En un turno, los agentes realizan sus acciones, una sola por cada agente, y modifican el medio sin que este varíe a no ser que cambie por una acción de los agentes. En el siguiente, el ambiente puede variar. Si es el momento de cambio del ambiente, ocurre primero el cambio natural del ambiente y luego la variación aleatoria.

En una unidad de tiempo ocurren el turno del agente y el turno de cambio del ambiente. Los elementos que pueden existir en el ambiente son obstáculos, suciedad, niños, el corral y los agentes que son llamados Robots de Casa.

1.1 Modelado e Implementación

El modelo empleado en la simulación, se baso en 3 módulos principalmente *commons*, *environment* y *agents*. En el modulo de ambiente se encuentra una implementación de *House* que no es mas que el ambiente en donde se desarrolla la simulación, contiene todas las funcionalidades necesarias para construir un campo aleatorio y manejar los distintos cambios que los objetos realizan, así como hacer efectivas las acciones de los agentes en el terreno.

Para construir un terreno es posible especificar la cantidad de obstáculos (o en por ciento), la cantidad de suciedad inicial (o en por ciento), la cantidad de bebes, un modelo para crear a los objetos de tipo bebe, un modelo de agente (con el cual se generan agentes), el tamaño entrado como tupla ($Dimension(cols, rows)$), el tiempo t en el que demora en variar el ambiente.

La clase *House* también se encarga de gestionar la variación que da lugar en el tablero cada t momentos, donde asumí como variación cambiar a los chicos que están fuera de los corrales, recolocar los corrales en nuevas posiciones(adyacentes entre si, de igual forma), reposicionar las suciedades, cambiar el lugar del o los agentes asi como los obstaculos, todo esto aleatoriamente.

En el método *turn_cycle* se encuentra el ciclo principal de nuestra simulación donde se le especifica si se dese paso a poso y si quiere una salida verbosa, en este ciclo como dice su nombre se suceden los turnos y el objetivo del Robot de Casa es mantener la casa (a.k.a el ambiente) limpia. Se considera la casa limpia si no más del *60 %* de las casillas vacías están sucias. Se sabe que si la casa llega al *60 %* de casillas sucias el Robot es despedido e inmediatamente cesa la simulación. Si el Robot ubica a todos los niños en el corral y el *100 %* de las casillas están limpias también cesa la simulación. Estos son llamados estados finales. Debe programar el comportamiento del robot por cada turno así como las posibles variaciones del ambiente. En este ciclo compruebo cada una de las condiciones anteriores antes de continuar al siguiente turno.

Luego está las implementaciones en el modelo *commons*, que refieren a lo común usado entre *environment* y *agents*. En este módulo encontramos las implementaciones útiles para:

- *Directions*: referido a la lista de direcciones nombradas, se puede obtener con *rdirs* una lista desordenada de direcciones, así como también encapsula otras funcionalidades necesarias relacionadas.
- *Coordinates* (**NamedTuple**): referido a la posición en el piso de la casa, con un método *on_direction* para obtener la correspondiente posición en la dirección especificada.
- *CellContent* (**Enum**): referido a los contenido de celdas, *Dirty*, *Empty*, *Obstácle* y *Playpen*.
- *Dimensions* (**NamedTuple**): referido a la dimensión del tablero.
- *distance* (**función**): función que describe la distancia, de forma que a partir del parámetro nom-

brado *extended* se escoge uno u otro sistema de medición de distancias.

- *SimulationResult* (**NamedTuple**): tipo para almacenar de una manera estructurada la información obtenida de una simulación concluida.

- *AgentAction*(**Enum**) y *ChildAction*(**Enum**): para enumerar las posibles acciones que se pueden realizar y que en caso de escoger un acción el ambiente ejecuta dicha acción sobre el agente/chico en acción.

Por último tenemos el módulo de agentes o *agents* que contiene las implementaciones para estos dada diferentes estrategías, percepción, etc. Todo agente hereda del tipo *HouseAgent* que contiene implementaciones útiles para sus descendientes pero no posee una implementación precisa del método empleado por el ambiente para cuestionar la acción a realizar en el turno actual, el método *Action*. Este básicamente contiene un campo *carrying* y un campo *coord* con la posición en la casa asociada.

Tipos Concretos de agentes:

DummyRobot: se mueve aleatoriamente (con doble paso si es posible) esto con ninguna influencia del ambiente, recoge al chico si está en la celda actual y deja al chico si se encuentra en la casilla de corral, así como también recoge basura si esta en el camino errático.

Como idea principal para este *Robot* está la baja percepción del ambiente que de alguna forma solo observa lo que tiene a su alcance, (suponiendo que no recibe toda la información del ambiente), no tiene estrategia fija o por lo menos a largo plazo, más allá del siguiente paso.

FocusedRobot: esta ya es una implementación más avanzada de robot limpiador, este tiene una especie de objetivo o *goal* y hasta que no llegue a su objetivo, no realiza otra tarea que no esté relacionada con el consecuente objetivo(Avanza solo con un foco, si no tiene foco se queda en el lugar)

Para este último la idea pasa por una percepción completa sobre el ambiente, que traza estrategias a largo plazo según las condiciones del mismo y de si mismo.

2. Simulación

Los ambientes que utilizamos para la obtención de alguna medida de efectividad de las implementaciones realizadas de agentes, fueron obtenidos a partir de los siguientes argumentos:

Di Dimensión del escenario.

Nc Número de niños en la casa.

D% Por ciento de casillas sucias iniciales.

O% Por ciento de obstáculos iniciales.

T Tiempo cada cuanto se realizan las variaciones de ambiente.(donde el tiempo de la simulación es $T * 100$)

Los siguientes fueron los ambientes que tuve en cuenta para la obtención de resultados concluyentes sobre la eficacia de uno y otro modelo de robot.

Amb	*Di*	*Nc(%)*	*D(%)*	*O(%)*	*T*
A01	(10, 10)	3	0	0	10
A02	(15, 15)	2	15	5	16
A03	(20, 20)	1	25	2	11
A04	(15, 15)	6	0	0	12
A05	(20, 20)	6	12	5	12
A06	(20, 20)	1	5	20	14
A07	(10, 10)	1	20	5	14
A08	(15, 15)	4	2	0	10
A09	(20, 20)	4	3	1	14
A10S	(25, 25)	5	5	4	15

2.1 Resultados Obtenidos

Para un total de 30 corridas , por cada uno de los escenarios o ambientes distintos, a continuación los resultados concluyentes para cada ambiente y por cada implementación de agente, se mostrara en la tabla las tablas la cantidad de veces que cada uno completo la limpieza, mantuvo hasta el final un ambiente estable o fue despedido fallando en la limpieza, por otro lado el promedio de casillas sucias y la conclusión final por ambiente, de si constituye en promedio un despido o no:

C : Completado

S : Estable

F : Fallo (o despido)

FC : Conclusión Final (resultado a partir de la cantidad de ambientes fallados, estabilizados y completados)

DummyRobot

Agente ShortPerception (DummyRobot):

Amb	*C(%)*	*S(%)*	*F(%)*	*FC*
A01	0.0	0.0	100.0	F
A02	0.0	40.0	60.0	F
A03	0.0	93.4	6.6	S
A04	0.0	0.0	100.0	F
A05	0.0	0.0	100.0	F
A06	0.0	96.7	3.3	
A07	76.6	10.1	13.3	C
A08	0.0	0.0	100.0	F
A09	0.0	0.0	100.0	F
A10	0.0	0.0	100.0	F

Promedio de casillas sucias:

Amb	*PCS*
A01	60
A02	87
A03	148
A04	135
A05	230
A06	85
A07	8
A08	136
A09	240
A10	364

Para la implementación con escasa percepción se obtuvieron los datos y esperados en la mayoría de los casos se tiene un despido o ambiente estabilizado, pero es poco probable completar un tablero con ninguna tarea o actitud para hacerlo.

FocusedRobot

Agente LongTermPerception (FocusedRobot:)

Amb	*C(%)*	*S(%)*	*F(%)*	*FC*
A01	96.6	0.0	3.4.0	C
A02	100.0	0.0	0.0	C
A03	100.0	0.0	0.0	C
A04	40.0	0.0	60.0	F
A05	60.0	0.0	40.0	C
A06	100.0	0.0	0.0	C
A07	100.0	0.0	0.0	C
A08	86.6.0	0.0	13.4	C
A09	100.0	0.0	0.0	C
A10	100.0	0.0	0.0	C

Promedio de casillas sucias:

Amb	*PCS*
A01	2
A02	0
A03	0
A04	81
A05	92
A06	0
A07	0
A08	18
A09	0
A10	0

Como podemos ver para este caso de un Robot con completa percepción sobre el ambiente y con una tarea de especificada, a cada paso se acerca al objetivo. Dado esto es fácil notar como en la mayoría de los momentos de cada escenario se llega a un caso completado, en una pequeña parte se culmina habiendo fallado y casi imperceptible el caso en que el ambiente se mantiene estable, dado que en cambio al anterior este robot reacciona según las condiciones de cada uno de los momentos y toma la mejor decisión(según su heurística).