Proyecto
Simulación y Programación Declarativa
Agentes

Colectivo de Simulación

Ariel Antonio Huerta Martín C412

1. Orden del Problema Asignado

El ambiente en el cual intervienen los agentes es discreto y tiene la forma de un rectángulo de N × M. El ambiente es de información completa, por tanto todos los agentes conocen toda la información sobre el agente. El ambiente puede varíar 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. A continuación se precisan las características de los elementos del ambiente:

Obstáculos: estos ocupan una unica casilla en el ambiente. Ellos pueden ser movidos, empujándolos, por los niños, una única casilla. El Robot de Casa sin embargo no puede moverlo. No pueden ser movidos ninguna de las casillas ocupadas por cualquier otro elemento del ambiente.

Suciedad: la suciedad es por cada casilla del ambiente. Solo puede aparecer en casillas que previamente estuvieron vacías. Esta, o aparece en el estado inicial o es creada por los niños.

Corral: el corral ocupa casillas adyacentes en número igual al del total de niños presentes en el ambiente. El corral no puede moverse. En una casilla del corral solo puede coexistir un niño. En una casilla del corral, que esté vacía, puede entrar un

robot. En una misma casilla del corral pueden coexistir un niño y un robot solo si el robot lo carga, o si acaba de dejar al niño.

Niño: los niños ocupan solo una casilla. Ellos en el turno del ambiente se mueven, si es posible (si la casilla no está ocupada: no tiene suciedad, no está el corral, no hay un Robot de Casa), y aleatoriamente (puede que no ocurra movimiento), a una de las casilla adyacentes. Si esa casilla está ocupada por un obstáculo este es empujado por el niño, si en la dirección hay más de un obstáculo, entonces se desplazan todos. Si el obstáculo está en una posición donde no puede ser empujado y el niño lo intenta, entonces el obstáculo no se mueve y el niño ocupa la misma posición.

Los niños son los responsables de que aparezca la suciedad. Si en una cuadrícula de 3 por 3 hay un solo niño, entonces, luego de que él se mueva aleatoriamente, una de las casillas de la cuadrícula anterior que esté vacía puede haber sido ensuciada. Si hay dos niños se pueden ensuciar hasta 3. Si hay tres niños o más pueden resultar sucias hasta 6.

Los niños cuando están en una casilla del corral, ni se mueven ni ensucian.

Si un niño es capturado por un Robot de Casa tampoco se mueve ni ensucia.

Robot de Casa: El Robot de Casa se encarga de limpiar y de controlar a los niños. El Robot se mueve a una de las casillas adyacentes, las que decida. Solo se mueve una casilla sino carga un niño. Si carga un niño pude moverse hasta dos casillas consecutivas.

También puede realizar las acciones de limpiar y cargar niños. Si se mueve a una casilla con suciedad, en el próximo turno puede decidir limpiar o moverse. Si se mueve a una casilla donde está un niño, inmediatamente lo carga. En ese momento, coexisten en la casilla Robot y niño.

Si se mueve a una casilla del corral que está vacía, y carga un niño, puede decidir si lo deja esta casilla o se sigue moviendo. El Robot puede dejar al niño que carga en cualquier casilla. En ese momento cesa el movimiento del Robot en el turno, y coexisten hasta el próximo turno, en la misma casilla, Robot y niño.

Objetivos

El objetivo del Robot de Casa es mantener la casa limpia. Se considera la casa limpia si el 60 % de las casillas vacías no están sucias.

2. Principales Ideas seguidas para la solución del problema

Para la inicialización del ambiente, dado la cantidad de filas, columnas, robots, niños, y obstáculos, primero se genera una posición aleatoria en el tablero donde se posiciona un corral, luego a partir de este, de forma aleatoria, siempre que este en los límites del tablero y que no esté ocupada, se pone otro corral y se repite el procedimiento con este nuevo corral hasta que estén colocados la misma cantidad de corrales que de niños. Una vez colocados los corrales, se pone de forma aleatoria en celdas válidas, los niños, los obstáculos, los robots y la suciedad.

Para la simulación de los niños, se buscan todos los niños que se puedan mover, que son aquellos que no están en un corral o que no están cargados por un robot o que no están rodeados por churre, y se decide por cada uno de forma aleatoria si se va a mover o no. En caso de moverse, se busca las celdas a las cuales puede moverse (vacías, o con obstáculos siempre que este pueda ser movido) y de forma aleatoria se decide una, en caso de que hubiera en dicha celda un obstáculo, se mueve este y los demás obstáculos en caso de que haya más de uno adyacentes en la dirección que se movió el niño; si la celda a la que se mueve está vacía, se comprueba cuantos niños hay en la cuadrícula de 3X3 en la cuál el niño es el centro y se buscan cuantas casillas están vacías en dicha cuadrícula, se calcula el menor número entre la cantidad máxima de suciedad que se puede generar y las cantidad de casillas vacías y se genera un número random entre 0 y dicho número previamente calculado que nos da la cantidad de suciedad generada, la cual de forma aleatoria se coloca en las casillas vacías. Los niños se mueven cada *t* unidades de tiempo.

Para la simulación de los robots de casa, se hizo empleo de un *bfs* que nos devolvía una matriz de distancias, a partir de la cual se buscaba los caminos más cortos hacia donde se dirigía los robots, lo cual ayudaba a decidir sus tareas. Para la decisión de la próxima acción a ejecutar del robot se usaba el estado actual en el que se encontraba, dependiendo del tipo de agente que era podía un estado implicar diferente acción para cada agente. Para mover al robot, como ya se explicó se calcula la matriz de distancias, con dicha matriz, se busca según el tipo de agente la siguiente acción, si se está con un niño cargado, sería buscar el corral más cercano, y a partir de esa posición ir buscando por la matriz de los valores del *bfs* la posición adyacente de menor valor y hacer el mismo proceso con esa nueva posición, hasta llegar a una posición con valor 2 o 1, que significa que el robot que estamos analizando puede moverse allí ya que con un niño en brazos puede dar hasta dos pasos; de igual manera se realiza para cuando no carga un niño, la diferencia estaría en que se busca la suciedad o el niño más cercano con el *bfs*, y se busca a partir de donde el robot quiere ir, el camino por la matriz del *bfs* hasta llegar a una posición de valor 1, en la cual movemos al robot. Siempre, antes de decidir buscar a un niño se pregunta si es posible llegar a un corral, de no ser posible, la acción sería acercarse a una suciedad para limpiarla, ya que no se podría dejar al niño en el corral. Lo descrito anteriormente se realiza por cada robot. El robot se mueve cada una unidad de tiempo.

3. Modelos de Agentes considerados

Para la solución del problema se consideraron dos modelos de agente, el reactivo y el pro-activo.

Primeramente, se consideró un agente reactivo, del cual conocemos que debe ser capaz de percibir su ambiente y responder de un modo oportuno a los cambios que ocurren para lograr sus objetivos. En cada turno, dicho agente dada la información del ambiente, decide en caso de no cargar niño, entre ir a la suciedad más cercana a limpiarla o ir a cargar a niño más cercano (siempre dándole prioridad a cargar niños y

llevarlos al corral ya que tenerlos sueltos puede entorpecer la tarea del robot de mantener limpia la casa), en caso de cargar un niño se busca el corral más cercano en el cual dejarlo. Este agente se vería menos afectado ante las variaciones del ambiente debido a que en caso de haber cambiado el ambiente se podrá adaptar con la mejor decisión posible solo con comenzar el próximo turno.

El otro agente considerado fue el pro-activo del cual sabemos que debe ser capaz de mostrar un comportamiento dirigido a objetivos tomando la iniciativa para lograr sus objetivos. En la solución, debido a lo explicado previamente acerca de como los niños al dejar suciedad entorpecen el cumplimiento de los objetivos del robot, entonces este agente tiene un comportamiento dirigido a dejar a todos los niños en el corral primero y luego limpiar la suciedad. Dicho comportamiento se mantendría durante varios turnos hasta que se cumpla la meta (dejar a todos los niños en el corral) o se descubra la imposibilidad de esta (hay un o más corrales a los que no se puede acceder).

4. Ideas seguidas para la implementación

Para la implementación del ambiente, se creo un datatype Elements que representa los elementos del ambiente, cada uno siendo un data constructor, donde todos poseen su posición en el ambiente, y en el caso del robot y del niño además se cuenta con un booleano que nos dice si el robot está cargando un niño y si el niño está siendo cargado por un robot, respectivamente.

La representación del ambiente es una lista de Elements, donde en ella solo se encuentran los elementos del ambiente, es decir, la suciedad, los obstáculos, los niños, los robots y los corrales, las casillas vacías no se guardan.

Dada una posición, con el método index se devuelve una lista de todos los elementos que estén en dicha posición. Este método sirvió de mucho puesto que haciendo uso del pattern matching, se podía como en el caso del robot, simularlo como una máquina de estados, por ejemplo, en el caso del agente reactivo, si index devuelve en su posición al mismo robot y a una suciedad, el robot limpia dicha casilla, si devuelve una casilla vacía, se busca entonces la próxima posición, si devuelve un niño, un robot y un corral donde el niño esta siendo cargado por el robot, entonces se procede a dejar al niño, y de forma similar con los distintos estados. Este método ayudó también a determinar los niños que se encuentran en un corral y ya no pueden moverse, entre otros muchos usos.

Para el agente pro-activo, en el caso de estar en una casilla con suciedad, se comprueba si existe algún niño que pueda ser llevado a un corral, donde también el corral sea accesible, de poder realizar esto, no se limpia la suciedad y se mueve el robot; en caso contrario, se limpia la suciedad.

La mayoría de las funciones son puras, exceptuando aquellas que trabajan con randomIO y la función de simulación que imprime en la consola.

Para el *bfs* se utilizo la clase Data.Matrix, con la cual era más sencillo el manejo de un array bidimensional, en dicha matriz se colocaban los valores de distancia una vez realizado el *bfs.* La idea seguida para el bfs es empezando de la posición inicial, analizar los adyacentes, en caso de ser posiciones válidas, asignarles el valor de la distancia con respecto a la posición inicial, y guardarlos en una lista para que luego sean analizados y así recursivamente hasta que todas las posiciones hayan sido visitadas. A las posiciones no válidas se les asigna un valor (-2), estas posiciones son las que no se pueden alcanzar, ya sea un obstáculo o un corral con un niño, entre otras.

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

A partir de la ejecución de las simulaciones del problema se puede apreciar que a medida que aumenta el tamaño del ambiente, los obstáculos y los niños, se muestra más la ventaja del modelo reactivo en situaciones variables con respecto al modelo pro-activo. Por lo que se puede concluir, que el modelo reactivo es el que ofrece una mejor solución al problema descrito.