Universidad de La Habana
Facultad de Matemática y Computación

Proyecto Final de
Programación Declarativa
y
Simulación

Gabriel Fernando Martín Fernández C411

Marco General

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

Obstáculos: estos ocupan una única 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 aparezla 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 adyacentee, 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.

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

Principales ideas

Ya que los robots son los encargados de mantener la limpieza, estos serán los agentes que emplearemos para resolver la tarea dada. El ambiente en el cual se desenvuelven estos agentes es discreto, dinámico, determinista y de información completa. Basado en estas características es necesario que los robots sigan un conjunto de reglas de decisión que tengan en cuenta la situación completa del ambiente y a su vez sean capaces de reaccionar a cambios fuera de su control.

Cada vez que el ambiente cambia todos los niños del tablero que estén fuera de un corral o no sean cargados por un robot pueden generar suciedad. Debido a esto una prioridad inicial es tratar de disminuir la cantidad de niños que generan suciedad. Si los robots priorizaran la limpieza entonces en el tiempo que demoran en trasladarse de una suciedad a otra un niño que pudiesen haber cargado puede haber generado varias casillas sucias. Además, la cantidad de casillas ensuciables es finita por lo que el peor caso de seguir la estrategia de lidiar con los niños primero está bien definido pero en el caso de lidiar con la suciedad primero pudiese ser que se llegase a un punto donde esta se genera siempre que el robot la limpia.

Ya sabemos que harán nuestros robots que estén solos pero el cómo también es importante. En este caso como el objetivo inicial de estos robots es disminuir la cantidad de niños que generan suciedad lo qué harán será dirigirse al niño más cercano que tengan y posteriormente llevarlo a un corral. Por supuesto la tarea de limpiar no queda olvidada y en el caso de que no se puede llegar a un niño que esté suelto el robot se dedica a limpiar, lo cual también incluye el caso donde no queden más niños sueltos.

Otro aspecto a considerar es el comportamiento de los robots cuando cargan un niño. Como en este caso se sigue la prioridad de disminuir los niños sueltos la primero será llevarlos al corral más cercano lo más rápido posible para dedicarse a buscar otro niño suelto. Un inconveniente de esto es que si no hay más niños accesibles realmente sería más eficiente limpiar con este robot pero esto también pudiese provocar que al cambiar el ambiente otro niño sea accesible y

sea necesario entonces que este robot lleve al niño al corral y despúes se dirija al niño ahora accesible perdiendo turnos donde este genera suciedad. Decidir entre cual de estos dos casos es algo complicado y se optó por la solución más sencilla: priorizar llevar al niño al corral y después seguir con la limpieza.

Modelos de Agentes considerados

Teniendo en cuenta lo expuesto anteriormente se ve la necesidad de tener agentes proactivos que sigan ciertos objetivos concretos y estén orientados a su cumplimiento. Además estos agentes deben ser capaces de reaccionar a cambios en el medio por lo que también deben ser reactivos. En este sentido consideramos la implementación de nuestros agentes con estas cualidades como esenciales.

Agente reactivo (implementado)

Quizás el más simple que cumple nuestros requerimientos. Estos agentes tratarán de cumplir su objetivo siguiendo ciertas reglas y tendrán en cuenta la característica dinámica del medio. Una carencia de este modelo de agentes para el problema dado es que no tienen en cuenta una comunicación entre los distintos robots y varios agentes pueden dedicarse al mismo objetivo. Esto puede provocar retrasos en el objetivo general de mantener la limpieza y es la principal carencia que presenta el empleo de este tipo de agentes. De esto concluimos que es importante otra caracterísitca para nuestros agentes: deben ser sociables

Agente Inteligente (implementado)

Los agentes inteligentes serán bastante similares a los agentes reactivos a la hora de resolver el problema pero con la importante diferencia de la comunicación entre ellos. Esto les permite realizar múltiples tareas sin estorbarse y de esa forma completar el objetivo más rápidamente. Un problema de estos agentes es que si bien no se estorban entre ellos al elegir las tareas esto no implica que estas sean distribuídas de la forma más eficiente posible.

Sistema multiagente

Probablemente el acercamiento más eficiente al problema, pues las decisiones se tienen en cuenta de manera coordinada entre múltiples agentes y teniendo el cuenta el mejor resultado para el objetivo dado. La ventaja de este sistema está en su enfoque en optimizar la tarea general de todos los agentes y no simplemente optimizar la tarea de cada agente por si solo. Usualmente conllevan el empleo de una IA distribuída para la toma de decisiones.

Detalles de la Implementación

Para la implementación de la simulación del problema descrito se empleó el lenguaje de programación funcional Haskell. La simulación es manejada desde el módulo *Simulation* el cual inicializa al módulo del ambiente (*Enviroment*) y se encarga de hacer que este realize los cambios cada turno. El módulo del ambiente genera las posiciones iniciales de manera aleatoria y adémas crea una cantidad pertinente y también aleatoria de suciedad y obstáculos al inicio. También este módulo determina cuando sea el turno pertinente para que sea necesario un cambio aleatorio y emplea el módulo *Kids* para generar dichos cambios. Además se llama al módulo *Robots* para que se encargue de generar los cambios en los agentes en todos los turnos (antes de los cambios

aleatorios). Los otros módulos implementados sirven de apoyo a las funciones principales de la aplicación y a continuación describimos el contenido de los ficheros y sus módulos (los cuáles tienen el mismo nombre que sus respectivos ficheros).

Ficheros

Main.hs: Fichero desde donde se inicializa la aplicación.

Simulation.hs: Se encarga de inicializar la simulación, ejecutar los turnos de esta e imprimir los detalles relevantes.

Eviroment.hs: Inicializa el estado del ambiente y se encarga del llamar a las funciones encargadas de generar los cambios ya sea por el efecto de los niños o los robots.

Kids.hs: Maneja todo lo relacionado con el accionar de los niños en el mapa cuando el ambiente cambia.

Robots.hs: Se encarga del funcionamiento de los robots y los cambios que estos generan en el ambiente.

Bfs.hs: Métodos y funciones necesarias para el bfs y otros métodos empleados por los robots.

Tools.hs: Métodos diversos empleados en la aplicación principalmente para el trabajo con listas y matrices.

Representations.hs: Contiene la representación de algunos de los datos usados internamente.

Visualization.hs: Métodos para visualizar los datos y generar la sencilla interfaz visual de la consola.

Ejecución

Para entrar en el entorno se debe ejecutar el fichero *Main.hs*:

```
$ ghci Main.hs
```

Se inicializa la aplicación llamando a la función main la cual requiere 8 parámetros enteros:

n: tamaño de la primera dimensión del ambiente.

m: tamaño de la segunda dimensión del ambiente.

t: tiempo entre cambios aleatorios del ambiente.

kidsCount: cantidad de niños.

robotCount: cantidad de robots.

robotType: tipo de robot, 0 para el reactivo y 1 para el inteligente.

seed: semilla, si es 0 se utiliza IO unsafe para generar distribuciones aleatorias cada vez que se llame a la función main de lo contrario se emplea el valor como semilla para generar los valores aleatorios del problema, para una misma semilla se obtendrá una misma distribución

turnsCount: cantidad de turnos a simular.

Ejemplo:

```
Enviroments> main 5 5 1 2 3 1 0 10
```

Al principio de la ejecución se imprime la leyenda de la interfaz que se presenta en la consola para visualizar el problema. Luego de pasar la cantidad de turnos especificada se da información adicional relevante respecto a algunos datos de interés del problema. Si se desea seguir la ejecución de un mismo problema que se terminó en un turno dado basta con que ese problema se haya ejecutado con una semilla personalizada y es posible simplemente llamar a main nuevamente con parámetros similares pero con un contador de turnos mayor para ver los que ocurre en los turnos siguientes.

Consideraciones

Para probar las diferencias entre los resultados de ambos gentes se generó un ambiente de dimensiones 10x10, t=2, con 15 niños y 5 agentes y se procedió a simular 100 turnos. Este experimentó se repitió para distintos valores de la semilla y para cada tipo de agente. Se obtuvieron los siguientes resultados luego de 5 casos distintos:

Simulación 1

Agente reactivo: Average dirt: 9.961851 % Maximum dirt: 30.69307 %

Agente inteligente: Average dirt: 11.57319 % Maximum dirt: 30.69307 %

Simulación 2

Agente reactivo: Average dirt: 4.815279 % Maximum dirt: 19.801981 %

Agente inteligente: Average dirt: 3.2233548 % Maximum dirt: 16.831682 %

Simulación 3

Agente reactivo: Average dirt: 7.3115897 % Maximum dirt: 31.683168 %

Agente inteligente: Average dirt: 7.2145205 % Maximum dirt: 31.683168 %

Simulación 4

Agente reactivo: Average dirt: 15.991844 % Maximum dirt: 50.495052 %

Agente inteligente: Average dirt: 12.691518 % Maximum dirt: 50.495052 %

Simulación 5

Agente reactivo: Average dirt: 1.2336439 % Maximum dirt: 9.9009905 %

Agente inteligente: Average dirt: 1.155989 % Maximum dirt: 9.0 %

Como se puede apreciar existe una diferencia entre ambos, el agente inteligente en general logra mantener el límite superior del porciento de suciedad del ambiente más bajo y también logra que la media de la suciedad del ambiente durante la simulación sea menor. En los casos donde se observa un mejor desempeño del agente reactivo en estos valores se debe al problema mencionado de estos agentes inteligentes a la hora de reservar objetivos pues no se asegura que la distribución de estos sea óptima y puede haber casos donde esto tenga un impacto sensible en el resultado (si la distribución es particularmente mala). Otro aspecto observado en varios casos probados es que incluso aunque el agente reactivo supere al inteligente en los valores de la media de suciedad y el máximo de suciedad alcanzado los agentes inteligentes en practicamente todos los casos terminan de limpiar primero el ambiente precisamente por el gran efecto negativo de la falta de coordinación de los agentes reactivos cuando hay pocos objetivos.

https://github.com/PhantomJoker07/CleaningBotsSim