Proyecto de Simulación y Programación Declarativa : Agentes

Osmany Pérez Rodríguez C - 412

Orden del problema asignado

Marco general

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 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 1 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 este 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.

Principales ideas seguidas para la solucion del problema

El proceso de la simulación comienza por crear un ambiente (Environment) a partir de las condiciones impuestas por el usuario al problema. Aquí se guardan todos los elementos del ambiente y este es el que recibe todas las modificaciones que se realizan en la propia simulación. La cantidad de dichos elementos (Children, Robots, Dirt, Obstacles, Playpen) está sujeta a la entrada que proponga el usuario en $Configuration.hs$. Tener presente que el resultado del proceso de simular depende en cierta medida de los parámetros de entrada que se impongan al problema, ya que por ejemplo una disparidad notable de Robots y Children puede provocar que la limpieza no se logre nunca.

Se conoce por la orientación que el ambiente varía aleatoriamente cada t unidades de tiempo. Esa variación es provocada por el accionar de los niños, y estos a su vez obedecen el comportamiento de moverse cada las ya mencionadas t unidades de tiempo. Estos a su vez generan de forma aleatoria suciedad (Dirt) .

Es el trabajo de los Robots limpiar la suciedad, pero estos obedecen cierto tipo de comportamiento, el cual también es definido a preferencia del usuario.

Modelos de agentes seleccionados

El tipo de comportamiento que obedecen los Robots está basado en dos tipos de agentes, y se define en el parámetro $simType$ de $Configuration.hs$.

Si el mismo es 0, los Robots tienen un pensamiento proactivo ya que primeramente buscan llegar a un niño(Child) que no esté dentro del corral (Playpen). Una vez que logra cargar a un niño busca la forma más cercana de llevarlo a un corral. Esto se realiza con un simple objetivo: imposibilitar que los niños puedan generar más suciedad. Durante este proceso limpiar la suciedad pasa a un segundo plano, hasta que los niños se pongan en una posición que les impida ensuciar. Ya llegado este punto, entonces se limpia toda la suciedad restante y concluye así cualquier cambio significativo realizado al ambiente.

Si el tipo de simulación es 1, los Robots tienen un pensamiento reactivo. En su accionar busca cualquier objetivo que represente una mejoría al ambiente actual. Se mueve en dirección al niño o suciedad más cercana que tenga. Si este logra cargar a un niño busca entonces el elemento más cercano entre la suciedad y un corral, y se mueve en su búsqueda. Esto se logra ya que se asume que el Robot tiene la habilidad de limpiar aún cuando tiene un niño encima. Además si

este se encuentra parado sobre una suciedad, decide limpiarla y después iniciar el proceso de búsqueda una vez más.

Estás búsquedas mencionadas se realizan con un algoritmo BFS y por esta razón siempre que se encuentre el elemento deseado, también se obtiene el camino más corto para su realización.

Implementación

Se implementó un proyecto utilizando $stack$ con la siguiente estructura:

- /app
  - Main.hs

    En dicho archivo se hace el llamado a la simulación y se imprime el ambiente en su estado final.
- /src
  - Configuration.hs

    ```
    simType :: Int
    simType = 0
    --simType = 0 Look for closest child and try to put it in a Playpen as a priority
    --simType = 1 Look for either child or dirt, and put it in Playpen and clean it respectively.

    dimension :: (Int, Int)
    dimension = (7,7)

    childCount :: Int
    childCount = 7

    robotCount :: Int
    robotCount = 3

    obstCount :: Int
    obstCount = 6

    dirtCount :: Int
    dirtCount = 2

    t :: Int
    t = 3

    totalTime :: Int
    totalTime =30
    ```

    En este archivo, el usuario debe definir las condiciones necesarias para la realización del experimento. A partir del mismo se construye el estado inicial del ambiente para después realizar la simulación. El archivo es un ejemplo de como se puede crear el mismo.
  - Board.hs

    ```
    type Coord = (Int, Int)
    ```

```
data Object
  = Child Coord
  | Obstacle Coord
  | Dirt (Maybe Object) Coord
  | Playpen (Maybe Object) Coord
  | Robot RobotType (Maybe Object) Coord
  deriving (Show,Eq)

data RobotType = AlphaRobot | BetaRobot
  deriving (Show,Eq)

data Environment = Environment
  {
    robots :: [Object],
    children :: [Object],
    dirts :: [Object],
    playpens :: [Object],
    obstacles :: [Object],
    time :: Int,
    dimension :: (Int,Int),
    seed :: Int
  }
  deriving (Show)
```

Aquí se define el ambiente y los elementos que pueden estar en el mismo. Además de varias funciones auxiliares que facilitan características específicas del ambiente para la realización de otros métodos.

- Behaviour.hs

  Principalmente se encuentran los métodos que hacen que cambie de estado el ambiente.

  - `childAction :: Object -> Environment -> Environment`

    Este método recibe como parámetro un niño y decide si lo mueve o no con la restricción de las t unidades de tiempo. También modela el movimiento aleatoria apoyándose en métodos auxiliares y la subsecuente generación de suciedad.

    Recibe además el ambiente actual y devuelve el ambiente modificado tras el accionar de dicho niño.

  - `robotActionClosestChild :: Object -> Environment -> Environment`

    Aquí se realizan los cambios que lleva a cabo el Robot que se pasa como parámetro sobre el ambiente; dadas las disímiles situaciones en las que este se pueda encontrar. Este es el método representativo del agente proactivo que siempre está en búsqueda de un niño que no esté en su corral e intenta lograr ponerlo en uno. Recibe también el ambiente actual y retorna el ambiente modificado tras su procedimiento.

  - `robotActionImprove :: Object -> Environment -> Environment`

Este sería entonces el método que modela el comportamiento del agente reactivo. Busca tanto una suciedad como un niño y a partir de ahí decide el paso a realizar. El cambio puede ser limpiar suciedad si el mismo está parado encima de una o moverse dependiendo de su posición en búsqueda de cualquier condición que mejore el estado del ambiente; o sea que en un futuro contribuya a la limpieza del mismo. Recibe también el ambiente actual y retorna al ambiente modificado tras su procedimiento.

- Simulation.hs

```
simulation :: Int -> [Environment]
simulation  simType = simulationAux simType [generateEnvironment]

simulationAux ::Int-> [Environment ] -> [Environment ]
```

El primer método se apoya en otro auxiliar que genera el ambiente dadas las condiciones que el usuario brinda para el experimento y ese se pasa como parámetro a $simulationAux$ que es el que envuelve el procesamiento general del programa. Mientras el tiempo no sea mayor que el tiempo total definido, se hace un llamada a accionar tanto de cada niño como de cada robot, y en consecuencia pueden generar cambios en el ambiente. Una vez concluido esto se guarda el ambiente actual en una lista y se mueve el tiempo actual. De esta forma una vez concluido el método se posee el estado del ambiente en cada instancia de tiempo.

- Random.hs

  Contiene un método que dada una semilla, devuelve un número random y es imprescindible para lograr el comportamiento aleatorio del ambiente.

Resultados

Después de realizar el proceso con varias entradas de parámetro, se puede llegar a la conclusión que de manera un poco sorprendente el agente reactivo logra con mayor efectividad controlar la suciedad de la casa, a pesar de no tener un objetivo totalmente. Estos resultados están sujetos a los casos que fueron testeados manualmente. Hay una gran posibilidad que para ciertos casos los resultados sean totalmente diferentes a los alcanzados hasta el momento.

Enlace al repositorio de Github:

(https://github.com/osmany-perez-1998/my-project)