Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Inteligencia Artificial

Proyecto Nº 2

A^*spibot: Agentes y Búsqueda

Segundo Cuatrimestre de 2008

A^*spibot es un proyecto para desarrollar un robot aspiradora que efectúe la limpieza del piso de una habitación. El prototipo a desarrollar, denominado A^*spibot (como el proyecto), será diseñado e implementado como un *agente* que percibe información de su *entorno* (la habitación) a través de sensores y actúa sobre él mediante efectores. Desde el punto de vista del agente, la habitación será un entorno no accesible, estático, discreto y determinístico. A^*spibot será capaz de desplazarse por el suelo de la habitación, percibir suciedad y aspirarla. Naturalmente, la habitación contendrá muebles y paredes que obstaculizarán su desplazamiento, de esta forma el agente deberá ser capaz de detectarlos para luego eludirlos. Además, A^* spibot cuenta con una batería que le permite desempeñar sus funciones sin necesidad de estar enchufado a la pared mediante un cable, y tiene la capacidad de percibir el nivel de carga de la batería en todo momento. En general, una sola carga de batería no le será suficiente para realizar su tarea por completo, de forma que A^*spibot deberá dirigirse periódicamente hacia su posición de inicio, donde podrá recargarla y continuar. Deberá ser suficientemente precavido para no agotar su batería antes de arribar a la posición de recarga, o quedará varado en medio de la habitación y sin haber terminado su tarea.

A^*spibot deberá limpiar la habitación lo mejor posible, y con el mínimo consumo de energía. Por esta razón, entre otras capacidades de razonamiento, el agente es capaz de efectuar búsquedas A^* con el fin de hallar caminos óptimos (respecto al consumo de energía) para moverse por la habitación, y de ahí la estrella ('$*$') junto a la 'A' en su nombre.

Para el presente proyecto deberá **diseñar e implementar el programa de agente** que controlará a A^*spibot. Además deberá implementar un **simulador de entorno** que permita testear el programa de agente desarrollado. Como lenguaje de implementación tanto para el programa del agente como para el simulador de entorno se adoptará el lenguaje Prolog. Finalmente, deberá implementar una interfaz gráfica en Java que facilite el ingreso de los datos de entrada de la simulación y la interpretación de los resultados de la misma. Concretamente, la interfaz deberá ofrecer una representación visual de una habitación, permitiendo al usuario definir una configuración particular. Además permitirá ejecutar la simulación, mostrando visualmente la evolución paso a paso de la misma.

La habitación (entorno)

La habitación será representada mediante una grilla de n filas (numeradas de 0 a $n-1$) por m columnas (numeradas de 0 a $m-1$). Cada celda de la grilla pude ser de tipo *obstáculo*, *baldosa*, *alfombra* o ser la celda *home*. La celda home es la posición donde la aspiradora comienza su tarea, donde recarga energía y donde deberá guardarse al terminar. Una celda de baldosa, al igual que una de alfombra, puede estar limpia o sucia.

Observación: Por tratarse de una habitación, se asume que las celdas del borde son de tipo obstáculo (representando pared y puertas cerradas). En otras palabras, la superficie de la habitación estará delimitada por un contorno de obstáculos.

La aspiradora (agente)

En cada instante el agente se encuentra en una posición (celda) determinada, y al comenzar a funcionar se encuentra en la celda home. El agente es capaz de realizar una serie de acciones y cada vez que ejecuta una acción percibe del entorno a través de sus sensores.

Percepciones. El agente posee cuatro sensores, a través de los cuales percibe información del entorno:

- un *sensor de tacto*, que le indica si ha chocado con algún obstáculo (pared, mueble, etc.), consecuencia de la acción anterior.
- un *sensor de luz*, que le indica si hay suciedad (polvo) en la posición en la que se encuentra.
- un *sensor de superficie*, que le indica si la superficie de la celda actual es de alfombra.
- un *sensor de energía*, que le indica el nivel de carga actual de la batería.

Acciones. El agente cuenta con acciones para desplazarse por la habitación: *adelante*, *atrás*, *izquierda* y *derecha*. También cuenta con la acción *aspirar*, para limpiar la superficie del piso sobre la que se encuentra, y con la acción *apagar*, para detener su funcionamiento.

Metas. La meta del agente es limpiar la habitación, volver a la posición inicial (home) y apagarse.

Efectos de acciones (sobre el entorno)

El agente puede moverse hacia cualquiera de las cuatro posiciones adyacentes a la actual mediante las acciones adelante, atrás, izquierda y derecha. Mediante la acción adelante, el agente intentará desplazarse hacia la celda que se encuentra en la misma columna que la actual, pero en la fila anterior ($FilaActual - 1$), mediante la acción derecha, a la celda que se encuentra en la misma fila pero en la columna siguiente ($ColumnaActual + 1$), etc. Si la celda a la que intenta desplazarse el agente contiene un obstáculo, entonces no se desplaza (permanece en la posición en la que estaba) y su sensor de tacto le indicará que chocó.

El agente puede ejecutar la acción aspirar en cualquier momento. Si la celda sobre la que se encuentra el agente está sucia, entonces pasará a estar limpia luego de la ejecución de aspirar. Si estaba limpia, entonces el entorno permanecerá inalterado.

Cada acción efectuada por el agente consume energía de la batería. Una acción de movimiento (adelante, atrás, izquierda y derecha) consume 1 barra de energía si el agente se desplaza hacia una celda de baldosa o si se choca contra un obstáculo, mientras que consume 2 barras si se desplaza sobre una celda de alfombra. La acción aspirar consume 1 barra de energía y la acción apagar no consume energía en absoluto. Si el agente se desplaza sobre la celda home, su batería será cargada por completo, y así podrá continuar normalmente con su tarea. Sin embargo, si la energía se agota (el nivel de energía llega a 0) y las aspiradora no se encuentra sobre la celda home, simplemente dejará de funcionar.

Conocimiento del agente acerca del entorno

Inicialmente, el agente **no** conoce la configuración de la habitación (es decir, la ubicación de los obstáculos ni de las celdas de alfombra y baldosa), y por supuesto tampoco sabe qué celdas

están sucias y cuáles limpias. La única información acerca del entorno que el agente tiene inicialmente es la ubicación (coordenadas) de la celda home (donde inicia la simulación).

Aparte de este conocimiento inicial, el agente podrá obtener información acerca del entorno mediante las percepciones que le brindan sus sensores (tacto, luz, superficie y energía). De esta forma, mediante el conocimiento inicial y considerando las acciones de movimiento ejecutadas (y si tuvieron éxito o no) el agente será capaz de calcular su posición actual en la grilla en cada instante. Más aún, el agente podrá ir descubriendo la configuración de la grilla mediante el análisis de sus percepciones.

IMPORTANTE: notar que para el programa de agente que controlará a *A*spibot*, la batería forma parte del "entorno", y la única forma de conocer el nivel actual de batería es a través del sensor de batería.

Capacidades de Razonamiento: búsqueda A*

Como parte de su mecanismo de razonamiento, el agente podrá emplear cuando resulte apropiado el método de búsqueda A* con el propósito de hacer más eficiente su desplazamiento (en términos de consumo energético). Concretamente, el agente podrá emplear el método de búsqueda A* para hallar un camino óptimo en términos de consumo de energía, desde un origen (generalmente su posición actual) a un destino determinados.

Cuando el agente efectúe una búsqueda lo hará desde su posición actual, sin moverse. En otras palabras, la búsqueda se llevará a cabo por completo antes de ejecutar la próxima acción, y por lo tanto sólo podrá emplear conocimiento de la habitación recabado hasta el momento. Debido a que el agente desconoce el contenido de las celdas aún no exploradas, no puede predecir, en el instante actual, cúal será el costo de atravesarlas. Más aún, siquiera sabe si podrá atravesarlas (ya que podrían ser celdas de obstáculo). Por esta razón, el agente **sólo considerará** en la búsqueda **celdas de la habitación ya exploradas**.

Las búsquedas A* pueden emplearse para diversos propósitos, y la única restricción para su uso es que éste sea razonable (es decir, que pueda ser justificado). Por supuesto, como requerimiento del proyecto, el algoritmo A* debe ser empleado **al menos una vez** durante el desempeño del agente. Consideremos un ejemplo de aplicación de A*. Una vez terminada su tarea el agente podría utilizar A* para encontrar un camino de costo mínimo desde su posición actual a la posición home. Luego de encontrar el camino podrá seguirlo mediante la ejecución de acciones de movimiento, y así llegar a home con un consumo óptimo de energía. También podría emplear A* simplemente para saber cuánta energía le demandaría viajar desde una posición hasta otra. Esto puede ser de utilidad para planificar una recarga de batería "segura", evitando que se agote la energía antes de llegar al cargador.

CONVENCIÓN: Como heurística deberá emplearse la distancia de Manhattan entre la posición considerada y la posición destino.

Evaluación del desempeño del agente

Para evaluar el desempeño del agente en la tarea de limpiar la habitación se tendrán en cuenta tres aspectos:

- *grado de completitud* de la tarea de limpieza, es decir, qué fracción del total de celdas sucias fueron limpiadas. Este aspecto es capturado por el cociente $Limpiadas/Sucias$, donde *Limpiadas* es la cantidad de celdas de la habitación que el agente efectivamente limpió y *Sucias* es la cantidad de celdas sucias en la habitación al momento en que el agente comenzó a funcionar.

- *consumo energético*, capturado simplemente por la cantidad de barras de energía consumidas por el agente durante la simulación.

- *guardado*, es decir si el agente finalizó la simulación en la posición home, o lo hizo en cualquier otra posición (ya sea porque ejecutó la acción apagar, o porque se quedó sin batería). El valor que se asociará a este aspecto es binario (si/no).

Convenciones de Diseño e Implementación

Representación en PROLOG

Habitación

Utilizaremos el par `[F, C]` (lista de dos elementos) para hacer referencia a la celda o posición de la grilla ubicada en la fila `F` y columna `C`. La configuración de la habitación será representada mediante hechos de la forma `celda(Pos, Tipo, Estado)`, donde `Pos` representa una posición en la grilla, `Tipo` $\in \{obs, alf, bal, home\}$ y `Estado` $\in \{limpio, sucio, -\}$. El '-' en `Estado` se emplea para las celdas con obstáculos (ya que no tiene sentido representar el estado de limpieza de un obstáculo) y para la celda *home* (ya que se asume que *home* siempre está limpia).

Percepciones y Acciones

Las percepciones serán representadas mediante la siguiente estructura (lista) PROLOG:

`[Choco, Sucio, Alfombra, Energía]`

donde `Choco` es 1 si el sensor de choque detectó una colisión resultado de ejecutar la última acción, y 0 en caso contrario, `Sucio` es 1 si el sensor de luz detecta suciedad, y 0 en caso contrario, `Alfombra` es 1 si el sensor de superficie detecta que el piso es de alfombra, y 0 en caso contrario y finalmente `Energía` es un natural entre 0 y $CapacidadBateria$ indicando el nivel actual de batería detectado por el sensor de energía.

Las acciones serán representadas mediante las constantes PROLOG `adelante`, `atrás`, `izquierda`, `derecha`, `aspirar` y `apagar`.

Programa de Agente y Simulador de Entorno

La implementación del programa de agente consta de dos predicados. Por un lado, el predicado `aspibot_setup(+PosHome)`, cuyo propósito es efectuar tareas de inicialización, entre ellas almacenar en el estado interno del agente el conocimiento inicial acerca del entorno (posición de la celda home). Por otro lado, el predicado `aspibot(+Perc, -Acc)`, implementando el comportamiento del agente en respuesta a las percepciones recibidas del entorno.

La implementación del simulador de entorno consta de tres predicados:

1. `sim_setup(+CapacBatería)`, cuyo propósito es efectuar tareas de inicialización para el simulador, entre ellas definir la **capacidad** (carga máxima) **de la batería** de A^*spibot, que constituye un **parámetro de la simulación**.

   IMPORTANTE: aunque el usuario del simulador podría elegir cualquier valor para la capacidad de la batería, se adoptará $\lfloor n * m/2 \rfloor$ como **valor default**, donde n y m son las dimensiones de la grilla. La cátedra empleará principalmente este valor para efectuar las pruebas.

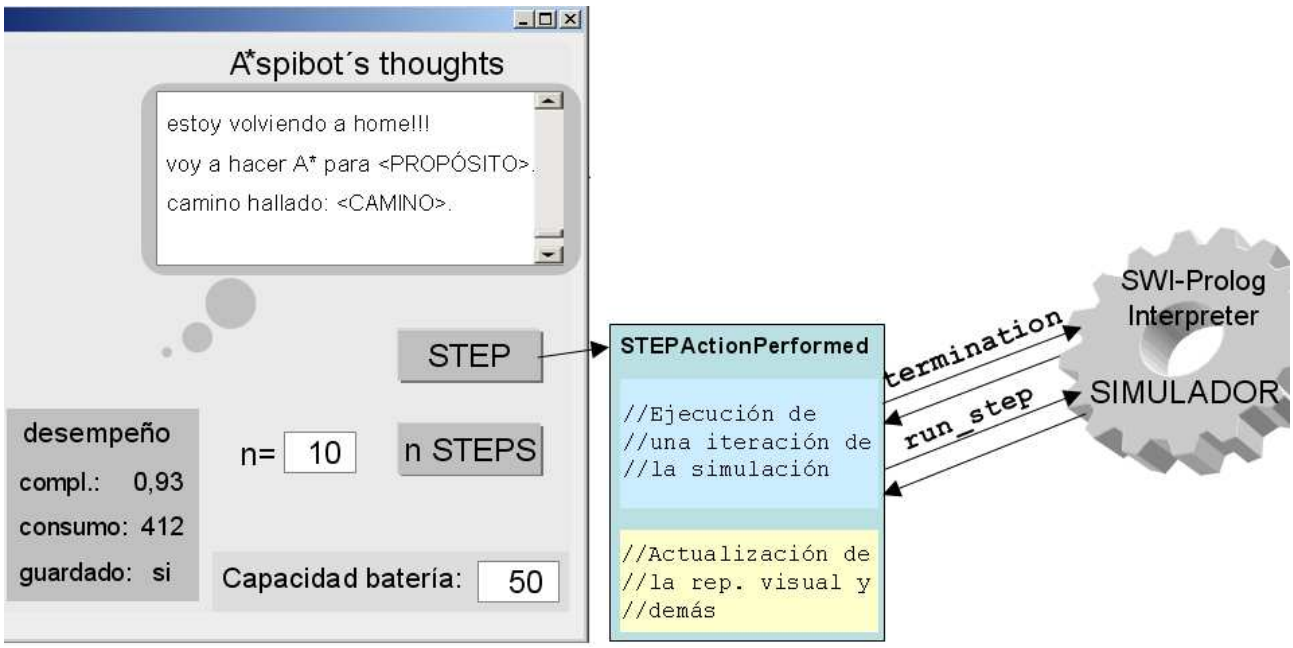

Figura 1: Interfaz Java y comunicación con simulador en PROLOG

2. `termination/0`, para determinar si se verifica la condición de terminación de la simulación.

3. `run_step/0`, para implementar un paso o iteración del ciclo de simulación. Concretamente, se obtiene la siguiente percepción que recibirá el agente, se invoca `aspibot/2` con la percepción, obteniendo la acción que el agente decidió realizar, y finalmente se actualiza el estado del simulador de entorno para reflejar la ejecución de la acción.

Interfaz Java y comunicación con simulador en Prolog

La interfaz permitirá ingresar en forma visual una configuración para la habitación e iniciará la simulación, invocando a los predicados `sim_setup/1` y `aspibot_setup/1`. Luego, mediante un botón **STEP**, permitirá la ejecución paso por paso de las simulación. Es decir, al hacer click sobre el botón STEP, se ejecutará un paso de la simulación (previa verificación de la condición de terminación), reflejando en la representación visual de la habitación los cambios que se produjeron en esta iteración, junto a cualquier otra información adicional que se desee mostrar relativa a la iteración. Mínimamente, deberá distinguirse celdas exploradas de las no exploradas hasta el momento, (por ejemplo, podrían pintarse los bordes de las celdas exploradas). La figura 1 muestra un esquema de la funcionalidad del botón STEP y la comunicación entre JAVA y PROLOG. La interfaz permitirá también ejecutar n pasos de la simulación mediante un botón **n_STEPS** (ver figura 1).

Al finalizar la simulación, deberá mostrarse los valores asociados a los tres aspectos de evaluación del desempeño (definidos en la sección *Evaluación del desempeño del agente*).

*A**spibot thoughts

Con el propósito de entender lo que el agente está haciendo (y "pensando") en cada instante de la simulación, se establece un esquema de comunicación entre el agente y la interfaz mediante el cual este primero "publica" sus pensamientos e intenciones. Concretamente, en cada iteración

de la simulación, el programa del agente puede hacer *asserts* de uno o más hechos de un predicado dinámico `thought/1`, donde el argumento de `thought` es un string con un pensamiento que el agente quiere publicar. Luego de cada iteración (step) de la simulación, se efectuarán retracts desde Java (más precisamente desde el propio método *STEPActionPerformed*) de todos los hechos `thought/1` "assertados" por el agente en dicha iteración, publicando los pensamientos en un área de texto con scroll, como muestra la figura 1.

El propósito de esta facilidad es brindar al usuario del simulador información necesaria para entender qué está ocurriendo en cada instante. Esto puede resultar de gran utilidad en la etapa de desarrollo y prueba del agente, pero también para la corrección del proyecto. Como **requerimiento** mínimo del proyecto, cada vez que el agente decida ejecutar el A* deberá publicarlo por esta vía, indicando (muy brevemente) el **propósito de la realización de la búsqueda** y fundamentalmente la **solución obtenida** (camino).

Condiciones de Entrega

Para el presente proyecto se establecen las siguientes condiciones de entrega:

1. Aparte de la implementación de la simulación, se deberá realizar un informe completo que documente las principales decisiones de diseño e implementación adoptadas al desarrollar tanto el entorno como el programa de agente, así como cualquier otra observación que se considere pertinente. Particularmente, debe quedar clara la estrategia que seguirá el agente para desarrollar su tarea. El informe debe incluir ejemplos de corridas de la simulación y también un listado de los principales programas implementados. Incluir una sección que explique detalladamente los pasos requeridos para ejecutar la interfaz suministrada.

2. Las comisiones pueden estar conformadas por hasta 2 integrantes (recordar que éstos deben estar previamente registrados con la cátedra).

3. La fecha límite de entrega del presente proyecto es el día **lunes 27 de Octubre de 2008** en el horario de clase. Los proyectos entregados fuera de término recibirán una penalización en su calificación, la cual será proporcional al retraso incurrido.

4. Deberá presentarse un folio plástico CERRADO (no entregar carpetas) conteniendo los siguientes elementos:

   - El informe impreso, que deberá estar encabezado por una carátula identificando claramente a los integrantes de la comisión.
   - Un disquete o cd conteniendo:
     a) Una carpeta `Prolog`, que contenga el código Prolog de los predicados implementados. En un archivo "entorno.pl" incluir todos los predicados que implementan el simulador de entorno. En un archivo "aspibot.pl" incluir todos los predicados que implementan al agente.
     b) Una carpeta `Java`, que contenga el código Java de la interfaz implementada.
     c) Una carpeta `Test`, que contenga todos los archivos requeridos para ejecutar la interfaz implementada, entre ellos el archivo ejecutable de la interfaz (.jar).

5. en la evaluación del proyecto se tendrá en cuenta que la implementación respete todas las convenciones establecidas. se recomienda prestar atención a dichas convenciones.