Inteligencia Artificial 2018-1
Proyecto 2: Búsqueda

IIMAS-PCIC

Luis Alejandro Lara Patiño
Roberto Monroy Argumedo
Alejandro Ehécatl Morales Huitrón

23 de noviembre de 2017

Índice

1. Funcionamiento del proyecto

1.1. Carga de información

Toda la información necesaria para el funcionamiento del programa debe cargarse en la base de conocimiento previamente a la ejecución. Las bases se guardan en el directorio `bases`, y se pasan como argumento al programa escribiendo el nombre de archivo, sin la extensión.

1.2. Ejecución

El punto de entrada al proyecto es el archivo main.pl. Este archivo está acondicionado para ejecutarse tanto desde el listener de Prolog como desde una terminal en sistemas operativos UNIX.

1.2.1. Desde una terminal UNIX

El archivo main.pl puede ejecutarse como cualquier otro *script*; solamente se debe navegar hasta el directorio del proyecto y desde allí ejecutar el comando:

```
./main.pl <nombre-base>
```

1.2.2. Desde el listener de Prolog

Asimismo, es posible ejecutar el programa desde el entrono de Prolog. Para ello, se debe consultar el archivo main.pl, y realizar la siguiente consulta:

```
?- main([<nombre-base>])
```

2. Aspectos interesantes del proyecto

- El módulo de toma de decisiones permite la recuperación grácil de un plan ejecutado a medias. Verifica cuáles objetivos han sido cumplidos ya e incorpora la información acerca de los objetos en sus manos.
- El algoritmo de búsqueda del módulo de planeación es multiobjetivo; considera varias de las decisiones a la vez para optimizar el plan generado. Antes de cumplir un objetivo, el módulo puede generar un plan que cumple parcialmente el objetivo siguiente, para economizar en el número de acciones y, por lo tanto, en el costo.

3. Estructura de la base de conocimiento

La estructura de base de conocimiento que utiliza este proyecto puede dividirse en tres secciones principales: conocimiento conceptual, conocimiento factual y conocimiento del mundo.

3.1. Conocimiento conceptual

En esta sección se almacena la información acerca de los objetos que participan en las actividades del robot. En el caso del asistente de supermercado, aquí se almacenan los productos de la tienda en forma de objetos, los cuales heredan las propiedades y relaciones de sus clases padre de acuerdo a la jerarquía implementada en el primer proyecto.

3.2. Conocimiento factual

En esta sección se almacenan las acciones realizables tanto por el empleado de la tienda como por el robot. También se mantienen aquí las decisiones que puede tomar el robot, así como la creencia que tiene el robot del mundo y las observaciones que éste adquiere durante la ejecución del programa.
Finalmente, se almacenan aquí los resultados de cada uno de los módulos del proyecto, dentro de los objetos `diagnostico`, `decision` y `agenda`.

3.3. Conocimiento del mundo

En esta sección se guarda la configuración que tiene el mundo del robot; las ubicaciones a las que éste puede acceder y qué objetos hay actualmente en ellas.
También existe un objeto `robot`, que mantiene los objetos que éste tiene en sus brazos y la posición en la que se encuentra.

4. Módulos

4.1. Simulador

La primera sección del proyecto que se ejecuta cuando éste inicia es el simulador. Éste se encarga de coordinar la ejecución de los tres módulos cuando sea necesario y de ejecutar el plan encontrado.
Al inicio del programa, el simulador ejecuta los tres módulos para generar un plan y cumplir las órdenes del cliente. En esta primera iteración, el robot confía plenamente en el reporte inicial del empleado de la tienda, y solamente busca entregar los productos que el cliente le pidió.
Una vez generado el plan, el simulador ejecuta las acciones secuencialmente. para determinar si una acción tuvo o no éxito, se extrae la probabilidad de éxito de la misma, almacenada en la base de conocimiento y mediante un generador de números aleatorios, se aplica dicha probabilidad. Adicionalmente, la acción buscar puede fallar si se recibe una observación que no concuerda con la creencia que tiene el robot del mundo en ese momento.
Si alguna de las acciones falla, el simulador vuelve a ejecutar los tres módulos del proyecto, con el fin de generar un nuevo plan para superar los obstáculos que pueda haber encontrado. Las acciones anteriores se repiten hasta que el robot cumple con todas sus encomiendas. En este momento, el programa termina.

4.2. Módulo de diagnóstico

El funcionamiento del módulo de diagnóstico se compone de dos partes principales, la asignación de productos dadas las observaciones y la búsqueda.

4.2.1. Asignación de productos

Cuando se ejecuta el módulo de diagnóstico por primera vez, no hay observaciones que presenten errores que encontró el robot. entonces el diagnóstico se reduce a encontrar la secuencia de acciones que (en teoría) llevó a cabo el almacenista para acomodar los productos que indicó en su reporte.
Si existe al menos una observación acerca de un estante en la base, se comparan los objetos del estante en cuestión con lo que existían en la creencia, los que sobran se acomodan aleatoriamente en los otros estantes (porque si no están en el lugar que deben, el almacenista los pudo colocar en cualquier otro lugar).
Una vez modificada la creencia con la información que nos dio la observación, se guarda en la base que se pasará como parámetro al módulo de toma de desicion.

4.2.2. Búsqueda

Para hacer una búsqueda se necesita la lista de estantes con los productos de cada uno (dados por la asignación de productos que se explico en el punto anterior) y una locación actual. También una lista de objetos y posiciones restantes. La forma en como se obtiene es realizando una búsqueda en la que en cada paso agregamos una acción ´´mover" o ´´colocar" hasta que la lista de objetos y posiciones restantes sea vacía en cuyo caso agregamos la locación inicial para así obtener (en una lista) el conjunto de acciones de final a principio que realizó el almacenista.
La forma de nuestra función sucesor toma en cuenta la minima falla que pudo cometer el almacenista, es decir suponemos que los productos están en su lugar.

4.3. Módulo de toma de decisión

Una vez que el módulo de diagnóstico ha actualizado la creencia de acuerdo a las observaciones, el módulo de decisión se encarga de revisar las órdenes del cliente y las inconsistencias en la tienda para generar objetivos a cumplir.

Cuadro 1: Disposición inicial de objetos.

Estante 1 (bebidas)	Estante 2 (comida)	Estante 3 (pan)
refresco		galletas
cerveza		cereal
sopa		

En primer lugar, este módulo revisa las órdenes del cliente, y crea objetivos para cumplirlas. A continuación, revisa los objetos en la creencia del robot para detectar cuáles están fuera de lugar, y crea objetivos para reacomodarlos.
Finalmente, el módulo de toma de decisión organiza los objetivos de manera que se le dé prioridad a las peticiones del cliente. Después de satisfacer las órdenes del cliente, el robot puede dedicarse a reordenar los objetos que estén fuera de lugar.
Ahora bien, puede suceder que este módulo sea ejecutado después de que el robot falló al ejecutar una acción, por lo que algunos objetivos podrían haberse cumplido, o el robot podría tener objetos en sus brazos. Para resolver estas situaciones, el módulo de toma de decisión verifica la lista de objetivos generados para ver si alguno ya se cumplió, y si es así, lo elimina de la lista. Adicionalmente, se crean objetivos de reacomodar para los objetos que el robot tiene en brazos. Como último paso de procesamiento, se eliminan aquellos objetivos que tienen una instancia de entregar y una de reacomodar con el mismo objeto, dejando sólo la de entregar.

4.3.1. Ejemplo de ejecución

Considérese el caso en el cual se tienen tres estantes; el primero de bebidas, el segundo de comida y por último el estante de pan. En dichos estantes se reparten los objetos que se muestran en el cuadro **??**. Además, el cliente ha pedido que se le entregue el refresco. Se observa que la mayoría de los objetos están en los estantes adecuados, con excepción de la sopa y el cereal, que deberían estar en el estante 2.
De acuerdo a esta información, el módulo de decisión intenta satisfacer primero las órdenes del cliente y establece el objetivo `entregar(refresco)`. Posteriormente, detectará que tanto la sopa como el cereal están fuera de lugar y creará los objetivos `reacomodar(sopa)` y `reacomodar(cereal)`. Finalmente, el módulo organizará el orden de los objetivos para que el robot pueda pasar por dos objetos que se encuentren en le mismo estante. En este caso, pondrá el objetivo `reacomodar(sopa)` inmediatamente después de `entregar(refresco)`.

4.4. Módulo de planeación

La entrada al módulo de planeación son las decisiones que se tomaron en el módulo anterior. Éstas son los objetivos que debe cumplir el plan generado por este módulo.
Para representar un camino en el árbol de búsqueda, se utilizó una estructura de datos llamada nodo, la cual tiene tres campos: el primero indica el estado inicial de dicho camino. El segundo campo consiste en una lista de acciones que forman el camino. Finalmente, el tercer campo contiene el estado final alcanzado tras seguir la secuencia de acciones desde el estado inicial.
Para codificar un estado, se empleó una lista. Dicha lista contiene la configuración del robot en ese instante, y listas para cada ubicación en el mundo, que indican los objetos presentes en cada una de ellas.
El algoritmo utilizado en este módulo es una versión modificada del algoritmo de búsqueda primero el mejor. Éste explora una gráfica utilizando dos conjuntos de nodos: los "blancos", o ya explorados, y los "grises", también llamados "frontera", que son aquellos nodos expandidos, pero que aún no han sido explorados. En cada iteración, este algoritmo toma el mejor nodo de los grises; en este caso, el que ofrece la mayor recompensa, y lo expande. Los sucesores expandidos se guardan en el conjunto de los grises, y

el nodo extraído se guarda en los blancos. Si el nodo seleccionado cumple alguno de los objetivos, dicho objetivo se elimina de la lista actual y se continúa la búsqueda para cumplir los restantes. El algoritmo termina cuando ya no queda ningún objetivo por cumplir.
El algoritmo implementado en este proyecto tiene algunas particularidades que es importante mencionar: En cada iteración del algoritmo, las acciones expandidas no corresponden a todas las acciones realizables por el robot, sino solamente aquellas aplicables al estado actual. Por ejemplo, la acción agarrar no será expandida si en la acción anterior no se buscó un objeto. La acción colocar será expandida solamente para objetos que el robot tenga en sus manos. De esta manera, se limita el árbol de búsqueda sólo a acciones completamente posibles, disminuyendo el tiempo necesario para encontrar un plan y la memoria utilizada.
El principal criterio para elegir un nodo es la recompensa que éste obtiene. El cálculo de la recompensa se basa en la última acción en el camino desde la raíz hasta el nodo en cuestión. Si la misma acción ya se realizó con los mismos argumentos, significa que posiblemente el plan esté atascado en un ciclo de acciones. Por lo tanto, no se recompensa este tipo de acciones. Además, si una misma acción contribuye a cumplir más de un objetivo, se incrementa la recompensa que se le asigna, dando mayor prioridad a los objetivos más inmediatos.

4.4.1. Ejemplo de ejecución

Para mostrar el funcionamiento de este módulo, se analizará el caso en el cual el módulo de decisión obtuvo el objetivo de entregar el objeto `refresco1` al cliente. Este objeto, como reportó el empleado, se encuentra en la ubicación llamada `e1`. El robot se encuentra actualmente en la posición `inicio`, y no trae nada en sus manos.
El árbol de búsqueda generado en este caso se muestra en la figura 1. En esta figura se muestra cada estado como un rectángulo que contiene cuatro círculos; uno por cada ubicación posible para el robot. Se marca con un círculo negro el objeto de interés, en este caso `refresco1`. La posición del robot se marca con un triángulo.
La raíz del árbol de búsqueda es el estado inicial, tal como se describió arriba. De acuerdo a las reglas establecidas en esta sección, desde este estado solamente es posible realizar la acción mover. No puede realizarse la acción agarrar, porque no se ha buscado ningún objeto; tampoco la acción buscar porque no hay objetos en la ubicación actual. La acción colocar es imposible porque el robot no tiene nada en sus brazos. Como hijos del estado inicial se muestran los tres estados resultantes de llevar a cabo la acción mover a cada una de las ubicaciones restantes.
En la siguiente iteración, el algoritmo debe elegir uno de los sucesores para ser expandido. Los dos últimos hijos no contribuyen a alcanzar el objetivo, por lo que no reciben recompensa. El único hijo que recibe recompensa es el primero, por lo que es el que se elige. Se observa que a partir de este estado es posible expandir nuevamente la acción mover. También es posible realizar la acción buscar, ya que hay objetos en la ubicación actual, según la creencia del robot. Se generará un sucesor por cada objeto que haya en la ubicación actual, pero solamente se muestra aquí una instancia, marcada con ícono de ojo.
En esta ocasión, dos acciones reciben recompensa: buscar y mover a inicio. Es incluso posible que mover a inicio reciba una mayor recompensa que buscar. Sin embargo, al elegir la primera, ninguno de sus sucesores recibe recompensa, por lo que se quedarán en la frontera hasta que todos los nodos grises tengan recompensa 0. Por lo tanto, la acción buscar será seleccionada antes que cualquier sucesor de mover a inicio.
Ya que en cualquier momento es posible expandir la acción mover, se omitieron dichos nodos por brevedad. En este caso es posible aplicar la acción agarrar, ya que el objeto se buscó en la acción inmediata anterior. Se observa que, en todos los sucesores de la acción agarrar, el robot ya tiene el objeto deseado en su posesión. Se muestran los sucesores que consisten en la acción mover, pero también es posible buscar otro objeto. Como esta última acción no contribuye a alcanzar el objetivo actual, no recibe recompensa.
En la siguiente iteración, la acción mover a inicio es la única que recibe recompensa. Podría objetarse que ésta ya se expandió anteriormente, y que no debería recibir recompensa, pero aquello ocurrió en otro

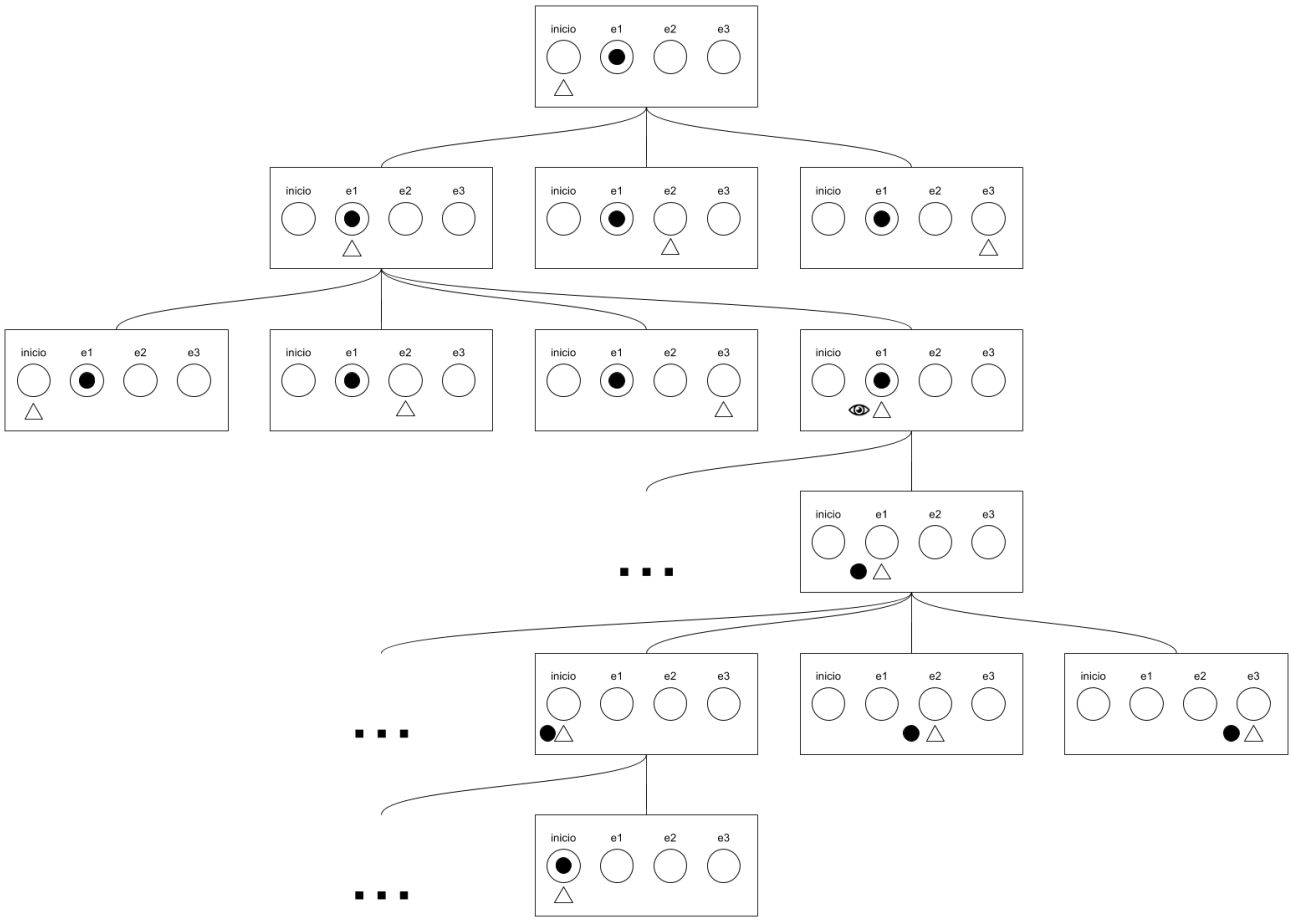

Figura 1: Árbol de búsqueda del módulo de planeación.

camino desde la raíz. En este camino en particular, es la primera ocasión en la que se expande la acción mover a inicio, por lo que sí recibe recompensa.
Una vez en la posición inicio, la única acción recompensada del robot es colocar el refresco en la posición de inicio, que en este caso significa la entrega al cliente.
Como este estado cumple el objetivo planteado y no existen más objetivos por cumplir, la búsqueda se detiene y se regresa el plan encontrado.

5. Evidencia empírica del funcionamiento del proyecto

A continuación probamos el proyecto con varios casos.

5.1. Ejemplos

5.1.1. Problema similar al *storyboard*

En este ejemplo trabajamos con tres estantes y los mismos productos del *storyboard* que se nos proporcionó.

- Reporte del almacenista:

```
e1 => [cerveza1,refresco1]
e2 => [sopa1,cereal1]
e3 => [galletas]
```

- Sin embargo el mundo real se encuentra de la siguiente manera:

```
e1 => [refresco1,cerveza1,sopa1]
e2 => [cereal1]
e3 => [galletas1]
```

- El cliente pide un refresco.
- La base de conocimiento de este ejemplo se encuentra en el directorio *bases* con el nombre *ejemplo* y la corrida del simulador en el directorio *docs/corridas*.

Resultado: En la ejecución hubo algunas acciones del robot no exitosas pero los tres módulos responden de forma adecuada en cad paso.

5.1.2. Problema con más objetos

En este ejemplo trabajamos con los mismos tres estantes pero ahora tenemos productos extra: jugo (bebida) y brownies (pan). Además, en la realidad, el robot se encuentra con un estante vacío.

- Reporte del almacenista:

```
e1 => [cerveza1,refresco1,jugo1]
e2 => [sopa1,cereal1]
e3 => [galletas1,brownies1]
```

- Sin embargo el mundo real se encuentra de la siguiente manera:

```
e1 => [refresco1,cerveza1,brownies1]
e2 => []
e3 => [cereal1,jugo1,galletas1,sopa1]
```

- El cliente pide una sopa.
- La base de conocimiento de este ejemplo se encuentra en el directorio *bases* con el nombre *ejemplo2* y la corrida del simulador en el directorio *docs/corridas*.

Resultado: La ejecución se realiza de manera exitosa.