Inteligencia Artificial
Planeamiento: Strips

Diego Marcovecchio (LU: 83815) Tomás Touceda (LU: 84024)

24 de Noviembre de 2010

Índice general

Capítulo 1

Introducción

1.1. Descripción

Este proyecto consiste en la implementación de un planificador STRIPS para un Mundo de Bloques similar al utilizado para los trabajos prácticos durante el cuatrimestre. El mundo posee un estado inicial fijo en el que todos los bloques se encuentran apoyados sobre la mesa. El objetivo del planificador es: dado un conjunto de metas, encontrar una secuencia de acciones que permita alcanzar todas ellas en un estado final.

1.2. Modularización

Dada la sencillez del planificador, a diferencia de los proyectos anteriores, toda la implementación del proyecto se encuentra en el archivo *strips.pl*. Más adelante en el informe se detallará cada uno de los predicados utilizados.

Capítulo 2

Estado del mundo

2.1. Relaciones entre bloques

El estado del mundo se describe mediante un conjunto de predicados que denotan relaciones entre bloques. Dichas relaciones son:

- *sobre(A, B)*: indica que el bloque A está inmediatamente arriba del bloque B.
- *libre(A)*: indica que el bloque A no tiene ningún bloque encima.
- *enMesa(A)*: es verdadera cuando el bloque A está apoyada encima de la mesa (es decir, no hay ningún bloque *debajo* de A).

2.2. Acciones

Las únicas acciones que pueden realizarse en este mundo son las acciones de apilar, y desapilar bloques. Presentamos, a continuación, el formato de dichas acciones en la notación de STRIPS:

- *apilar(A, B)*: coloca al bloque A inmediatamente arriba del bloque B

  Precondiciones

  ```
  [libre(A), libre(B), enMesa(A)]
  ```

  Add_List

  ```
  [sobre(A, B)]
  ```

  Del_List

  ```
  [libre(B), enMesa(A)]
  ```

- *desapilar(A, B)*: coloca el bloque A que está encima del bloque B sobre la mesa. Precondiciones

  ```
  [sobre(A, B), libre(A)]
  ```

  Add_list

  ```
  [enMesa(A), libre(B)]
  ```

  Del_list

  ```
  [sobre(A, B)]
  ```

Capítulo 3

Búsqueda

3.1. Características del algoritmo

En todos los casos, el algoritmo utilizado para encontrar caminos es A*.

En nuestra implementación, el algoritmo tiene una única meta; esto significa que para decidir cuál es el camino más corto entre un conjunto de n metas, se realizan n ejecuciones del algoritmo, y se comparan los costos de cada una, seleccionando finalmente la meta que resulte más barata. Una alternativa posible es implementar el algoritmo para buscar en un conjunto de metas, lo cual significaría cortar antes el proceso de búsqueda para metas más cercanas; el problema es que esta implementación, con muchas metas lejanas, resultaba en un costo de memoria mucho mayor al deseado, y la ganancia en velocidad de cómputo del camino no es tan grande (ambas posibilidades tienen el mismo orden de ejecución).

3.2. Representación de nodos

Los nodos en el algoritmo de búsqueda están representados por su posición en el mapa, el costo parcial asociado, el nodo antecesor y una dirección. La posición es una lista *[X, Y]* con las coordenadas; el nodo antecesor indica desde qué nodo se pasó al nodo actual; la dirección indica hacia qué punto cardinal estará mirando el agente, y el costo parcial asociado se calcula según el costo del nodo antecesor, más el costo de desplazamiento (1 si es *plain*, 2 si es *mountain*) más el costo de giro (0 si la dirección en el nodo es la misma que la dirección en el padre, 1 si es necesario realizar un giro de un nodo al siguiente). Nótese que es posible construir el path a un nodo concatenando recursivamente todos los antecesores.

Los únicos nodos que se generan son aquellas posiciones del mapa cuyo terreno es *plain* o *mountain*. De esta manera, los bosques, el agua, y los trozos de mapa no explorados no son considerados en la búsqueda. Esto convierte a nuestra implementación en un algoritmo pesimista, dado que asume que no se puede caminar por las partes desconocidas del mapa. Se encuentra definida, adicionalmente, una constante para indicar que el costo de un camino es «infinito», utilizada para indicar que un camino dado no puede llevar a la meta.

3.3. Función heurística

La función heurística empleada en nuestro A* es la *distancia manhattan* de un nodo a la meta. Dado que la meta es única, esta función no tiene ningún tipo de complicaciones, y garantiza subestimar el costo de movimiento.

3.4. Control de nodos visitados

Para evitar caer en ciclos y obtener siempre el camino minimal, el algoritmo utiliza el predicado dinámico *visitados/1*. Cada vez que se generan los vecinos de un nodo, se checkea si las posiciones de cada uno de ellos ya fueron visitadas o están en la frontera; en cualquiera de estos dos casos, si el costo del nodo visitado o en frontera es mayor al del camino actual, es borrado y reemplazado por el nuevo.

3.5. Resultado del algoritmo

De existir un camino por las partes ya exploradas del mapa hacia la meta deseada, el algoritmo siempre lo encuentra; el resultado de la aplicación de nuestro A* es una lista de nodos; en la práctica, como se verá más adelante, dicha lista se traduce a una secuencia de acciones a realizar.

En caso de que sea absolutamente necesario cruzar por una zona desconocida del mapa, el algoritmo devolverá un camino vacío y con el costo «infinito» definido por la constante mencionada anteriormente.

Capítulo 4

Estrategias

Dependiendo del momento, el estado y la percepción del mundo, Bugor puede utilizar diferentes estrategias de comportamiento para lograr sus intenciones (principalmente: no morir, conseguir oro, explorar y mejorar su fight_skill). El manejo de las estrategias en el agente es realizado mediante una pila: según sea necesario, se apila una estrategia, se trabaja de acuerdo a ella, y posteriormente se desapila y se continúa trabajando con la estrategia anterior. Cuando la pila de estrategias se encuentra vacía, por default se apila la estrategia de exploración.

Cada una de las estrategias tiene una prioridad definida, y únicamente se puede apilar una estrategia encima de otra cuando ésta tiene mayor o igual prioridad. Esta característica es la que indicará si una estrategia espera a que otra termine para comenzar a actuar, o si puede interrumpir el desarrollo de otra estrategia.

El agente además posee una pila de acciones, que corresponden siempre a la estrategia en el tope de la pila de estrategias. Bugor utilizará esta pila para planificar sus movimientos futuros dentro del mapa.

Pasaremos ahora a describir las estrategias implementadas.

4.1. General

La estrategia «general» (o «estrategia de exploración») es utilizada por Bugor para explorar pedazos de mapa que no conocía anteriormente. Sabiendo que el agente posee una representación interna del mapa del mundo, el primer paso es obtener todos los *nodos frontera* que sean *plain* o *mountain* en dicho mapa interno, como se muestra en la figura ¡¡¡¡¡¡¡FrontierNodes!!!!!!!!!!. Un nodo es considerado *frontera* cuando tiene al menos un vecino que no se conoce.

A continuación, todos estos nodos son ordenados según la *distancia manhattan* respecto a la posición actual de Bugor. El nodo frontera con la menor distancia manhattan es marcado como meta, y el agente se dirige hacia allí.

La decisión de ordenar los nodos heurísticamente y no elegirlos de manera aleatoria se tomó para evitar que el agente se dirija hacia una posición muy lejana por el simple hecho de explorar; resulta más provechoso explorar repetidas veces nodos cercanos en la frontera que explorar una única posición de la frontera muy lejana.

En el caso de que el nodo elegido no pueda ser accedido (es decir, cuando el costo de A* aplicado a ese nodo es la constante que representa *infinito*), se prosigue a explorar el siguiente nodo con menor heurística de la lista.

Mientras el agente esté aplicando esta estrategia, siempre que vea oro dentro de su campo visual, intentará ir a buscarlo inmediatamente; para ésto se apila la estrategia *Treasures* que se describe a continuación en la pila de estrategias.

Una vez que el mapa esté completamente explorado, si el agente recuerda la existencia de tesoros que no pudo recoger, coloca su meta de exploración en la posición de dichos tesoros. Para definir cuál de los

tesoros será buscado se realiza un balance entre la cantidad de turnos que pasaron desde la última vez que éste fue visto, y el costo de ir a buscarlo. Esto se diferencia de la estrategia Treasures en que únicamente ocurre cuando el mapa fue totalmente explorado, y en que los tesoros pueden estar en cualquier sector del mapa. La estrategia Treasures, en cambio, únicamente es utilizada cuando el agente divisa un tesoro dentro de su rango visual.

Por último, en caso de que el mapa ya esté completamente explorado y Bugor no recuerde la existencia de tesoros pendientes de ser recogidos, simplemente seleccionará locaciones al azar del mapa conocido y se desplazará hacia ellas (pudiendo, potencialmente, encontrar nuevos tesoros que antes no estaban, o buenas oportunidades para atacar agentes rivales).

4.2. Treasures

Esta estrategia es la encargada de definir el comportamiento de Bugor al recoger un tesoro.

El agente está diseñado para apilar esta estrategia **únicamente** cuando divisa un tesoro. Sabiendo esto, la estrategia de Bugor es buscar un camino a cada uno de los tesoros que se encuentran en su campo visual; estos tesoros se ordenan de acuerdo al costo del camino que se debe recorrer para ir a buscarlo.

Una vez que la lista de tesoros se encuentra ordenada, se toma el primero de los tesoros y se apila la secuencia de acciones a realizar para llegar a dicho tesoro en la pila de acciones, se realizan inmediatamente para intentar recogerlo, y se continúa explorando (es decir, se desapila la estrategia Treasures de la pila). Si todos los tesoros del campo visual de Bugor son, por el momento, inalcanzables, se prosigue explorando como se planeó originalmente y el tesoro queda igualmente recordado en la «memoria» del agente.

4.3. Attack

La estrategia de ataque es complicada de clasificar en el sentido de que no es considerada una estrategia en sí misma, pues en ningún momento es colocada en la pila de estrategias; lo que ocurre es que cuando la situación se presenta de manera favorable, Bugor golpeará a los agentes rivales para aumentar su fight_skill e intentar dejar a sus enemigos inconscientes. No está en la naturaleza de Bugor realizar planificaciones complejas para vencer a los rivales, si no que se limita a cumplir el resto de sus objetivos y atacar de manera conveniente para correr el menor riesgo posible, y obtener eventuales beneficios.

Una situación de ataque se considera favorable cuando el agente rival se encuentra dentro del rango de ataque de Bugor, pero *de espaldas* a éste. Concretamente, Bugor únicamente realizará ataques mientras esté fuera del campo visual de los agentes rivales, lo que le brindará la ventaja de ser el que comience la lucha y pegue la mayor cantidad de golpes (y, como se verá a continuación, una vez que el rival responda los ataques, Bugor escapará).

4.4. Flee

Bugor no es un agente particularmente agresivo, y si bien intentará combatir (como se detallará más adelante), la estrategia principal en caso de recibir un ataque es escapar lo más rápido y lejos posible, para minimizar el riesgo de quedar inconsciente.

Para realizar esto, inmediatamente después de recibir un ataque, Bugor apilará esta estrategia en la pila; dado que ésta es la estrategia con mayor prioridad (únicamente igualada por *Flee to hostel*), no podrá ser apropiada por ninguna de las anteriores. Una vez que esta estrategia está en el tope de la pila, Bugor se dirigirá de manera automática al punto más alejado que conozca en el mapa. Para realizar esto, se encuentran todas las locaciones del mapa con *plain* o *mountain*, y se ordenan heurísticamente de acuerdo a la *distancia manhattan* de la posición actual de Bugor. El punto con mayor valor heurístico se convierte en la nueva meta del agente, que realiza la búsqueda de un camino hasta dicha posición, lo traduce a una secuencia de acciones que es apilada en la estructura correspondiente, y las realiza sin

ningún tipo de interrupciones. En caso de que el punto con valor heurístico más alto no sea accesible momentáneamente (es decir, que haya que cruzar por una «zona negra» del mapa), se toma la siguiente posición más alejada y ésta se convierte en la nueva meta de búsqueda. Este procedimiento se repite hasta obtener un camino efectivo a algún punto lejano del mapa.

Una vez que finaliza la huída, el agente resume la estrategia anterior que estaba realizando (es decir, se desapila la estrategia Flee de la pila de estrategias). Nótese que el hecho de resumir la estrategia anterior no significa realizar exactamente las mismas acciones. A modo de ejemplo, si Bugor se encontraba buscando tesoros y fue atacado, huirá al punto más lejano posible en el mapa, y al finalizar continuará buscando tesoros. Dada la nueva locación del agente, muy probablemente haya otros tesoros más cercanos que el que se buscaba originalmente, así que las metas del agente cambiarán adecuadamente para buscar el tesoro más conveniente.

4.5. Flee to hostel

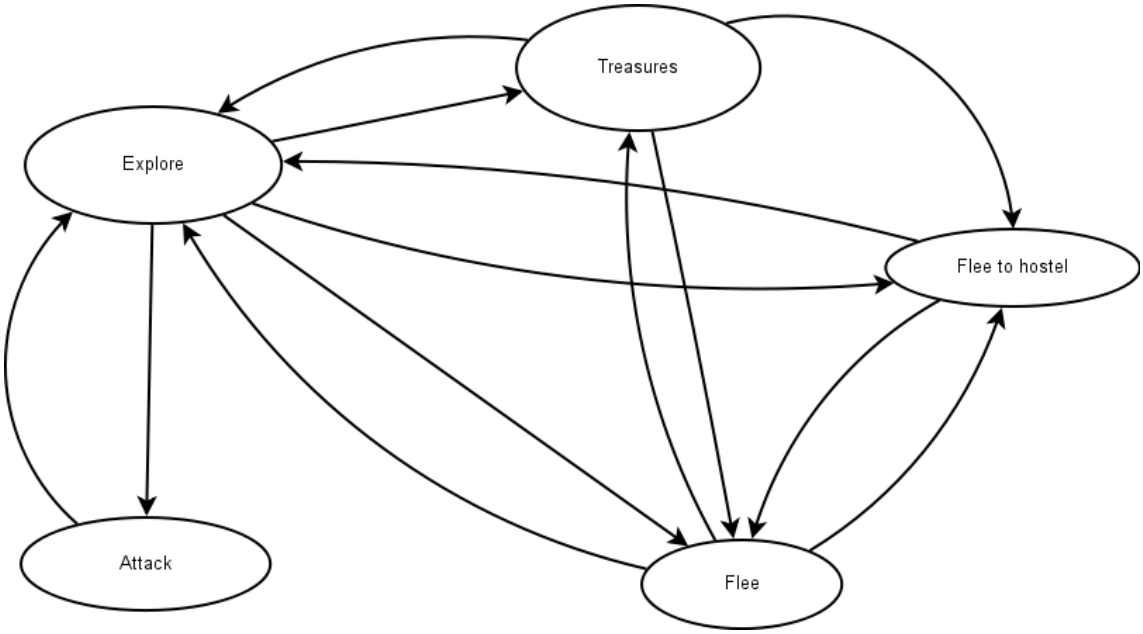
Esta estrategia es la que hace que Bugor se dirija a descansar de la manera más conveniente posible. Es invocada en cualquier situación en la que la stamina del agente sea menor al 40 % total, y posee la prioridad máxima (al igual que la strategia Flee). Esto significa que, en el caso de que Bugor sea atacado, si su stamina es **mayor** al 40 % total huirá como se dijo en la estrategia anterior; pero si la stamina resulta **menor** al 40 % total, entonces huirá hacia el hotel más cercano. Concretamente, dado este último caso la pila de estrategias puede contener una sucesión de estrategias *Flee-FleeToHostel-Flee-FleeToHostel...* en su tope. Si bien puede resultar estéticamente desagradable, ésto no resulta en un problema, porque el agente actúa de manera consistente y, una vez llegado a la meta, todas estas estrategias son desapiladas.

Flee to hostel actúa obteniendo primero todas las posadas que el agente conoce; para cada una de ellas, se busca un camino utilizando nuestra implementación de A*. Las posada que tenga un camino con el mínimo costo (es decir, la más cercana) es marcada como meta, y Bugor se dirige automáticamente hacia allí (y como se dijo anteriormente, dado que esta estrategia tiene la prioridad máxima, no podrá ser apropiada por ninguna otra). En caso de que ninguna de las posadas sea momentáneamente accesible, el agente vuelve a la fase de exploración para eliminar las «zonas negras» del mapa, e inmediatamente después de que alguna de las posadas sea accesible, se dirigirá a ella.

4.6. Grafo de transición

Para facilitar la comprensión, a continuación incluimos un grafo con las posibles transiciones entre estrategias, ejemplificando con una situación en la que podría ocurrir cada una.

Las transiciones están definidas de acuerdo a la prioridad mencionada para cada una de las estrategias. Para todos los casos en los que **no** se ejemplifica una transición, dicha transición ocurre cuando la estrategia *origen* finaliza y es desapilada de la pila de estrategias, dejando como nuevo tope a la estrategia *destino*.



- Explore ->Treasures: esta transición ocurre inmediatamente después de que Bugor encuentre un tesoro en su campo visual.
- Explore ->Flee to hostel: ocurre en cualquier caso en el que el agente esté explorando y su stamina actual sea menos del 40 % de su stamina máxima.
- Explore ->Flee: ocurre inmediatamente después de que Bugor reciba un ataque.
- Explore ->Attack: ocurre cuando Bugor ve un agente rival en su campo visual; Bugor evaluará si es conveniente atacarlo, o continuar explorando sin provocar conflictos.
- Treasures ->Flee to hostel: si el agente se está dirigiendo a recoger un tesoro y se da cuenta de que su stamina bajó a menos del 40 % del total, priorizará descansar para evitar quedar inconsciente. Nótese que no hay una transición *de vuelta*, porque una vez que el agente haya descansado, volverá a la estrategia de exploración (en donde reevaluará hacia qué sector del mapa dirigirse para recoger tesoros).
- Treasures ->Flee: si el agente está dirigiéndose a recoger un tesoro pero es atacado, se escapará inmediatamente.
- Flee ->Flee to hostel: si el agente está escapando (porque fue golpeado) y su stamina es menor al 40 %, entonces se dirigirá al hotel más cercano.
- Flee to hostel ->Flee: si el agente estaba dirigiéndose a un hotel y es atacado, inmediatamente intenta alejarse; luego de unas pocas acciones, Bugor volverá a intentar dirigirse hacia el hotel, con la esperanza de que, si el agente rival continúa al acecho en esa zona, no sea lo suficientemente rápido para atacarlo de vuelta.