TEMA 2. BUSQUEDA SIN INFORMACION

2.1 Agentes resolventes-problemas

Una vez que se nos formula un problema hay unos pasos que se deben realizar para llegar a resolver o solucionar ese problema. Lo primero de todo es **formular el objetivo**, siendo ese objetivo el conjunto de estados que llegan a satisfacer ese objetivo, siendo la tarea del agente buscar la secuencia de acciones que permiten llegar a un estado objetivo, el siguiente paso es la **formulación del problema**, la cual se basa en la decisión de que acciones y estados se tiene que tomar en cuenta para poder resolver el problema de una forma eficiente, a continuación, se realiza la **búsqueda**, la cual consiste en considerara que secuencia de acciones es la mejor para poder llegar a un **solución**, la cual será una secuencia de acciones, por último se daría la **ejecución** de la secuencia de acciones devuelta en el proceso de búsqueda.

En un sistema de **lazo abierto** no se tienen en cuenta otros "imprevisto" los cuales puedan arruinar a hacer más complicada la solución del problema, es decir, el agente tomaría las decisiones sin pensar en otras variables.

Un **problema** puede estar definido por cuatro componentes:

- *Estado inicial*: donde comienza el agente
- *Función sucesor*: conjunto de pares <acción, sucesor> los cuales nos dan los "pasos a seguir" y que estado se conecta con otro (una lista enlazada por así decirlo). Este junto al estado inicial definen el espacio de estados que es el conjunto de todos los estados alcanzables desde el estado inicial, dando diferentes caminos que son un conjunto de estados conectados mediante acciones.

  {<Ir(Sibiu), En(Sibiu)>, <Ir(Timisoara), En(Timisoara)>, <Ir(Zerind), En(Zerind)>}
- *Test objetivo*: se determinará cuales estados son objetivos, es decir, cuales de los estados nos llevan a la resolución del problema.
- *Costo del camino*: asigna una medida de rendimiento (km, dinero…) mediante la cual es pueda elegir de una forma rápida y eficiente, el camino que nos lleve hacia la solución.

  Todos estos elementos crean una o varias soluciones, las cuales se miden y clasifican en función del costo de camino, siendo la solución óptima la que menos costo tenga.

La **abstracción** se usa a la hora de formular los problemas para poder llegar a una solución, en donde tenemos que obviar todos los detalles que no sean relevantes para la resolución del problema.

Además hay varios tipos de problemas; por un lado tenemos los **problemas de juguete**, los cuales se utilizan para ejercitar o ilustrar los métodos de resolución de problemas y se pueden describir de una forma exacta y concisa ya que no pueden haber más variables (aspiradora, 8puzzle, problema 8-reinas…), por otro lado tenemos los **problemas del mundo real**, en los cuales la gente se interesa y le preocupan sus soluciones, además no solo tienen una descripción, por ellos siempre hay que generalizar al máximo sus formulaciones, es decir, pueden tener diferentes interpretaciones según el que formule el problema (redes, rutas de gps, estrategia militar…)

3.2 Estrategias de búsqueda no informada

3.2.1 Búsqueda en anchura

En esta estrategia **se expande primero el nodo raíz**, y a continuación se expanden todos sus **sucesores**, y a continuación sus sucesores, es decir, no se empieza por un nivel hasta que los nodos del anterior nivel estén completados. La lista de nodos ABIERTOS se gestiona mediante **FIFO.**

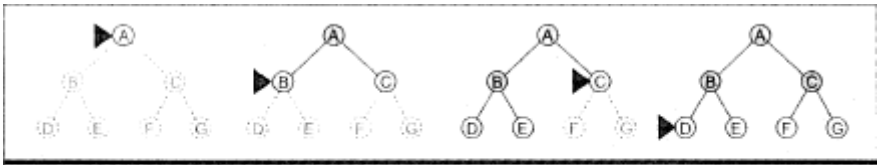

Figura 3.10 Búsqueda primero en anchura sobre un árbol binario sencillo. En cada etapa, el próximo nodo a expandir se indica con una marca.

3.2.2 Búsqueda en costo uniforme

Este método se aplica cuando los **costes** que hay entre los distintos nodos **no son constantes**, de modo que el gasto o coste no depende únicamente de cuantos nodos se recorran para llegar a la meta, si no también del coste. La lista ABIERTOS se trata como una lista de prioridades, en donde tienen mayor prioridad, el camino de la lista que menor gasto acumulado tenga (Dijkstra básicamente).

3.2.3 Búsqueda en profundidad

En este método siempre se **expande el nodo más profundo**, es decir, el nodo que está más a la izquierda del árbol y continúa hasta el nivel más profundo del árbol, donde los nodos no tienen sucesores. La lista de ABIERTOS se gestiona como una **pila LIFO**. El inconveniente es que puede haber una elección equivocada y hacer un camino muy largo, cuando mediante otro método se haría de una forma muy rápida. Hay variaciones sobre este método de búsqueda:

- *Profundidad acotada o limitada (L)*: sigue el mismo método que la búsqueda en profundidad, pero hasta un cierto límite L, a partir del cual no habría más sucesores, tiene un inconveniente ya que la meta o solución puede estar después de ese límite.
- *Profundidad iterativa*: se establece una sucesión de límite los cuales van aumentando hasta que se encuentra la solución, es decir, L=1>L=2>L=3…

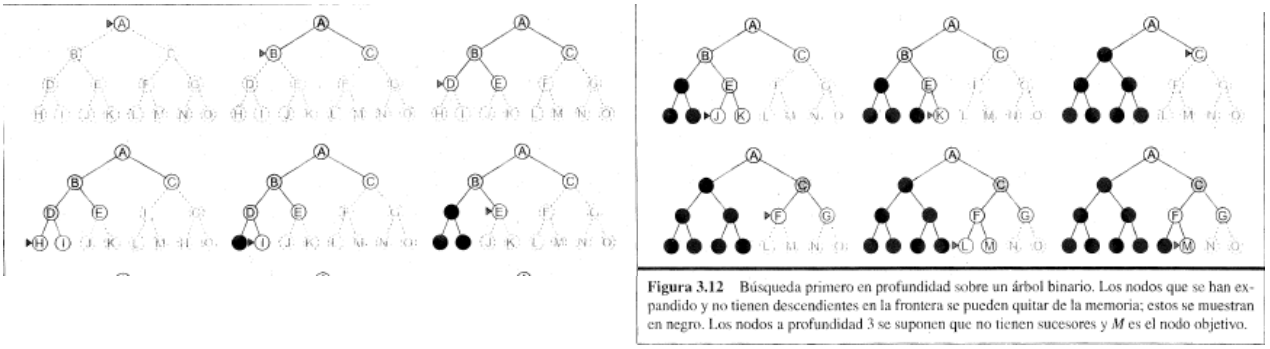

Figura 3.12 Búsqueda primero en profundidad sobre un árbol binario. Los nodos que se han expandido y no tienen descendientes en la frontera se pueden quitar de la memoria; estos se muestran en negro. Los nodos a profundidad 3 se suponen que no tienen sucesores y *M* es el nodo objetivo.

3.2.4 Búsqueda bidireccional

En este método se realiza una vez **conocemos la meta y el nodo raíz del árbol**, combinando dos métodos de búsqueda. Realizaremos el método de búsqueda desde el nodo meta y desde el nodo raíz para encontrar un nodo común el cual se reflejará en la lista ABIERTOS mostrándonos el camino desde el nodo raíz al nodo meta. Para poder economizar el gasto en memoria se pueden combinar los dos tipos de búsqueda anteriores

Para comprobar la eficacia de los métodos se estudian las siguientes características:

- *Completitud*: un método es completo si siempre encuentra solución, cuando esta existe.
- *Optimalidad*: cuando encuentra una solución, esta es la más cercana al estado de partida, sea en número de pasos, sea en coste acumulado de recorrerlos cuando este no es constante.
- *Complejidad espacial o en memoria*: dada por la estimación del tamaño de las listas a gestionar y mantener, en función del factor de ramificación y la profundidad de las soluciones. ¿Cuánta memoria se necesita para el funcionamiento de la búsqueda?
- *Complejidad temporal o en operaciones*: dada por la estimación de la cantidad de nodos o estados a generar y examinar, también en función del factor de ramificación y la profundidad de las soluciones. ¿Cuánto tiempo tarda en encontrar una solución?.

	Anchura	Profundidad	Profundidad limitada	Profundidad iterativa	Costo uniforme	Bidireccional
Completitud	Completo	No completo	No completo	Completo	Completo	Completo
Optimalidad	Óptimo	No óptimo	No óptimo	Casi óptimo	Óptimo	Óptimo
Coste espacial	$O(r^{p+1})$	$O(r*p)$	$O(r*p)$	$O(r*p)$	$O(r^{c+1})$	$O(r^{(p+1)/2})$
Coste temporal	$O(r^{p+1})$	$O(r^{p+1})$	$O(r^{p+1})$	$O(r^{p+2})$	$O(r^{c+1})$	$O(r^{(p+1)/2})$

r: factor de ramificación

p: profundidad de la mejor solución