Tema No. 2: Métodos de solución de problemas.
Actividad No. 19
Tipo de clase: Conferencia.
Título: Proceso inferencial. Búsquedas a ciegas.

Sumario:

- Espacio de estados.
- Estrategias de búsqueda. Criterios de evaluación.
- Estrategias de búsqueda a ciegas.

Objetivos:

- Determinar los elementos fundamentales de un proceso inferencial.
- Representar los problemas como espacios de estados. Definición, planteamiento y representación del problema.
- Valorar los diferentes métodos de búsqueda.
- Analizar diferentes algoritmos utilizados para resolver el problema de búsqueda en un espacio dado.

Bibliografía

[1] Patrick Henry Winston, Inteligencia Artificial, 3ra edición, 1992.
[2] Elaine Rich & Kevin Knight, Inteligencia Artificial, 2da edición, 1994.
[3] Dr. Rafael Bello Perez, Curso de Métodos de Solución de Problemas para la Inteligencia Artificial.

Introducción:

La solución de problemas puede considerarse una forma de razonamiento muy compleja que requiere la generación y asimilación de nuevas estructuras de memoria a fin de contestar una interrogante. Es la actividad mental de encontrar una solución a un problema. En el contexto del procesamiento de la información el enfoque dado a la solución de problemas ha sido tratar de trazar la gráfica de la secuencia de eventos desde la formulación del problema hasta su solución final; o sea, tratar de comprender el proceso que interviene para derivar una solución.

Los investigadores pioneros en I.A. tuvieron como su primer objetivo la solución de problemas que fueron difíciles de resolver mediante las técnicas computacionales existentes. Estos problemas generalmente no tienen solución algorítmica conocida o esta es tan compleja que no tiene una implementación práctica computacional.

La respuesta fue desarrollar nuevas técnicas de solución de problemas, similares a las humanas, una de las más importantes fue la búsqueda.

La búsqueda de la I.A. difiere de la búsqueda convencional sobre estructuras de datos esencialmente en que se busca en un espacio problema, no en una pieza de dato particular. Se busca un camino que conecte la descripción inicial del problema con una descripción del estado deseado para el problema, es decir, el problema resuelto. Este camino representa los pasos de solución del problema.

El proceso de buscar una solución a un problema produce un espacio solución, o sea, la parte del espacio problema que se examina realmente. A diferencia de las estructuras de datos que están

predefinidas y ya existen cuando comienza la búsqueda, los espacios problema son generalmente definidos proceduralmente, es decir, el espacio problema es creado a medida que es explorado. Se usan procedimientos para definir los siguientes estados posibles en el espacio a través de los cuales la búsqueda puede continuar desde el estado actual. Solamente los caminos explorados tienen que estar definidos explícitamente.

Hay diferentes alternativas para realizar la búsqueda. Desde un punto de vista podemos apreciar dos alternativas: **a ciegas y dirigidas**.

Con el siguiente ejemplo se ilustran estos. Supóngase que está en París sin un mapa y no habla francés ¿Cómo llegar hasta la torre Eiffel?

Una alternativa es seguir exhaustivamente cada calle de inicio a fin. Cuando se alcanza un final se busca una calle paralela y se sigue esta en dirección opuesta independientemente de si nos acercamos o alejamos del objetivo. Eventualmente esta variante consideraría todas las posiciones de nuestro espacio problema. Este tipo de búsqueda se llama a ciegas, ya que no usa conocimiento de cuán cerca estamos de la solución para tomar un determinado camino.

Alternativamente, podemos usar nuestro conocimiento sobre la torre para mejorar la eficiencia de la búsqueda. Suponiendo que el extremo superior de la torre puede ser visto desde cualquier lugar del espacio problema, podemos tomar la calle que nos parezca nos lleve en esa dirección. Esta es llamada una búsqueda dirigida. La búsqueda dirigida es la base de la I.A.

La búsqueda es un proceso de gran importancia en la resolución de problemas para los que no se dispone de técnicas más directas.

Definición formal de la solución de problemas en la IA.

Considérese un agente que actúa como resolvedor de problemas. Un problema es realmente una colección de información que el agente usará para decidir qué hacer.

Los elementos básicos de la definición de un problema son los estados y las acciones. Los elementos son los siguientes:

El **estado inicial** donde se encuentra el agente.

El conjunto de **acciones** posibles, disponibles al agente. El término operador se usa para denotar la descripción de una acción en términos de cuál estado será alcanzado ejecutando la acción en un estado particular.

El **espacio de estado** del problema es el conjunto de todos los estados alcanzables a partir del estado inicial mediante una secuencia de acciones cualquiera.

Un **camino** en el espacio de estado es una secuencia de acciones que conduce de un estado a otro.

El **criterio objetivo** es el criterio que el agente usa para aplicarlo a la descripción de un estado para determinar si este es un **estado objetivo** (lo que se desea obtener). Algunas veces hay un conjunto explícito de posibles objetivos y el criterio simplemente consiste en chequear si se ha alcanzado uno de ellos. Otra variante es especificar el objetivo por una propiedad abstracta, por ejemplo, el criterio de jaque mate del ajedrez.

El **costo de un camino** es una función que asigna un costo a un camino.

Una **solución** es un camino desde el estado inicial a un estado que satisface el criterio objetivo.

Otro concepto importante es el de **costo de la búsqueda** asociado con el tiempo y la memoria requisitos para encontrar una solución.

El **costo total** de la búsqueda es la suma del costo del camino y el costo de la búsqueda.

Ejemplos.
a) El rompecabezas de 8 piezas (8 – puzzle).

Consiste en un tablero de 9 casillas (3x3), en 8 de ellas se colocan fichas numeradas del 1 al 8 y una queda en blanco. El juego consiste en alcanzar una posición dada del tablero usando como regla que solo se puede mover una ficha a una casilla adyacente que esté vacía. La esencia del rompecabezas es dada cualquier numeración de las casillas obtener una con un orden específico, por ejemplo:

5	4	
6	1	8
7	3	2

➔

1	2	3
8		4
7	6	5

La formulación del problema es:

Estados: La descripción de un estado especifica la localización de cada ficha y la del espacio en blanco.

Operadores: Una ficha se mueve a la izquierda, derecha, arriba o abajo.

Criterio objetivo: Los 8 números quedan en un orden especificado.

Costo del camino: Cada paso cuesta una unidad.
Este problema a pesar de parecer simple tiene 9! = 36280 diferentes ordenamientos de los números en blanco, por lo que el método de simplemente generar estados y probar si cumplen el criterio objetivo lleva a una explosión combinatorial.

b) Misioneros y caníbales.

Tres misioneros y tres caníbales están en una ribera de un río junto con un bote que puede transportar uno o dos individuos. El problema consiste en llevar los 6 al otro lado sin que en ningún momento el número de caníbales supere al de misioneros.

Estados: se puede definir un estado como un terceto representando la cantidad de misioneros y caníbales en la ribera de salida y la ubicación del bote como tercer parámetro (+ en la ribera de salida y – en la de llegada). El estado inicial es (3, 3 ,+).

Operadores: Hay cinco operadores posibles: un misionero al bote, dos misioneros, un caníbal, dos caníbales o uno de cada uno al bote.

Criterio objetivo: estado (0, 0, -)

Costo del camino: cantidad de cruces.

Las figuras siguientes muestran cómo se puede ir generando nodos intermedios y un listado de los 32 nodos independientes de este problema. Como se puede apreciar de ellos hay algunos excluibles como (3,3,-) y (0,0,+) pues el bote no puede estar en un lado del río donde no hay nadie. También otros estados, como (2,3,+), que no satisfacen la restricción del problema. Luego de depurar de esta forma el espacio de estados nos quedan 18 estados a considerar.

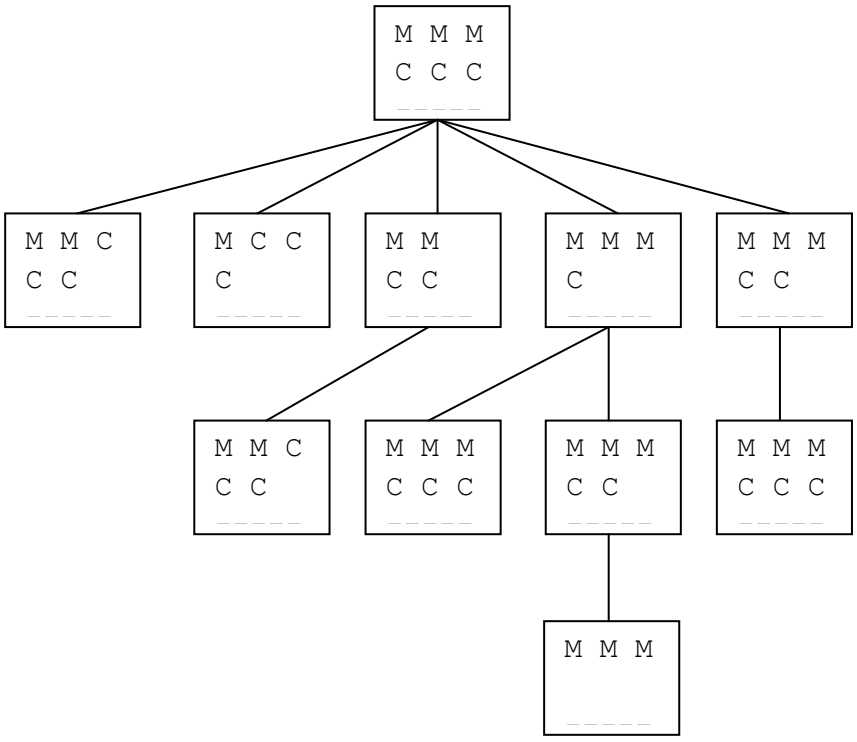


Figura 1 Representación del problema de los misioneros y los caníbales
Representación tabular del mismo problema.

(3,3,+)	(2,3,+)	(1,3,+)	(0,3,+)
(3,3,-)	(2,3,-)	(1,3,-)	(0,3,-)
(3,2,+)	(2,2,+)	(1,2,+)	(0,2,+)

```
(3,2,-)    (2,2,-)    (1,2,-)    (0,2,-)
(3,1,+)    (2,1,+)    (1,1,+)    (0,1,+)
(3,1,-)    (2,1,-)    (1,1,-)    (0,1,-)
(3,0,+)    (2,0,+)    (1,0,+)    (0,0,+)
(3,0,-)    (2,0,-)    (1,0,-)    (0,0,-)
```

Procedimiento general de búsqueda.

Dada la formulación de un problema la próxima acción es encontrar una solución, lo cual como ya se vio, consiste en generar nuevos estados a partir del estado donde se encuentra el agente. Este proceso se denomina expandir el estado. La expansión puede producir uno o varios nuevos estados. En el primer caso se toma este y se continúa, en el otro caso existen múltiples posibilidades para continuar la búsqueda por lo que es necesaria una selección.

Esta es la esencia de la búsqueda, seleccionar una opción y poner las otras alternativas a un lado para retomarla más tarde si la primera selección no conduce a una solución. La selección de cuál estado expandir primero se determina por la estrategia de búsqueda.

La búsqueda consiste en ir construyendo un espacio de búsqueda (la parte del espacio de estado que deja de ser abstracta). La raíz del espacio de búsqueda se corresponde con el estado inicial. Los nodos hojas del árbol corresponden a estados que no tienen sucesores en el árbol porque no han sido expandidos o porque no tienen sucesores. En cada paso el algoritmo de búsqueda selecciona un nodo hoja a expandir.

Debe distinguirse entre el espacio de estado y el espacio de búsqueda. Por ejemplo en el problema del viajero vendedor (considerando 20 ciudades) hay solamente 20 estados en el espacio de estado, uno por cada ciudad. Pero hay un número grande de caminos en el espacio de estado de modo que había una gran cantidad de nodos en el espacio de búsqueda.

Un algoritmo de búsqueda general (informal) es el siguiente:

```
function General_Search(Problem, Strategy) return Solucion;
      Inicializar el árbol de búsqueda usando el estado inicial o Fallar
      Loop do
         if no hay nodo que expandir then return Falla
         Seleccionar un nodo hoja a expandir de acuerdo a la estrategia.
         if el nodo contiene un estado objetivo
           then return Solucion
           else expandir el nodo y añadir los nodos resultantes al
                    espacio de búsqueda.
      end loop.
End
```

Estrategias de búsqueda.

Lo esencial en el campo de la búsqueda dentro de la IA es encontrar la estrategia correcta para un problema determinado. Para ello podemos evaluar una estrategia de acuerdo a los siguientes 4 criterios:

Completitud: ¿La estrategia garantiza encontrar una solución en caso de existir alguna?

Complejidad temporal: ¿Cuánto se demora en encontrar una solución?

Complejidad espacial: ¿Cuánta memoria necesita para realizar la búsqueda?

Optimalidad: ¿Encuentra la mejor solución entre todas las posibles?

Las estrategias sistemáticas que analizaremos estructuran el espacio de búsqueda en forma de árbol.

NO se parte del árbol en memoria.

Las complejidades temporal y espacial estarán referidas solo a la cantidad de nodos.

Consideraremos en una primera aproximación que la estrategia es óptima si la longitud del camino desde la raíz hasta el nodo objetivo alcanzado es mínima.

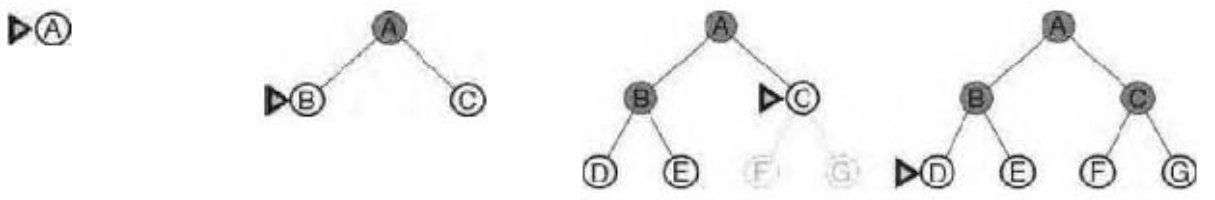
Se tienen los siguientes valores:
b (factor de ramificación): máximo número de hijos que tiene un nodo
m (profundidad): máxima longitud de un camino desde la raíz hasta una hoja
d: mínima longitud de un camino desde la raíz hasta un nodo objetivo

Estrategias de búsqueda a ciegas.

Las estrategias de búsqueda no informadas o búsquedas a ciegas son aquellas que no disponen de información sobre el número de pasos o sobre el costo del camino del estado actual hasta el estado objetivo, lo único que pueden hacer es distinguir un estado final de un estado no final.

Búsqueda en amplitud

En esta estrategia de búsqueda a ciegas, se expande primero el nodo raíz, luego se expanden todos los sucesores del nodo raíz, luego los sucesores de cada uno de ellos, y así sucesivamente, o sea, primero expande todos los nodos a un nivel de profundidad d para luego expandir los del nivel d + 1.De esta forma la búsqueda en amplitud analiza primero todos los caminos de longitud 1, después todos los de longitud 2, después todos los de 3, hasta llegar al estado objetivo, lo cual garantiza la optimalidad de dicha estrategia. La búsqueda en amplitud (Breadth First Search en inglés) es completa si el factor de ramificación b es finito. En la fig. siguiente se ve cómo se ejecuta el método sobre un árbol binario simple, después de 0,1,2 y 3 expansiones:



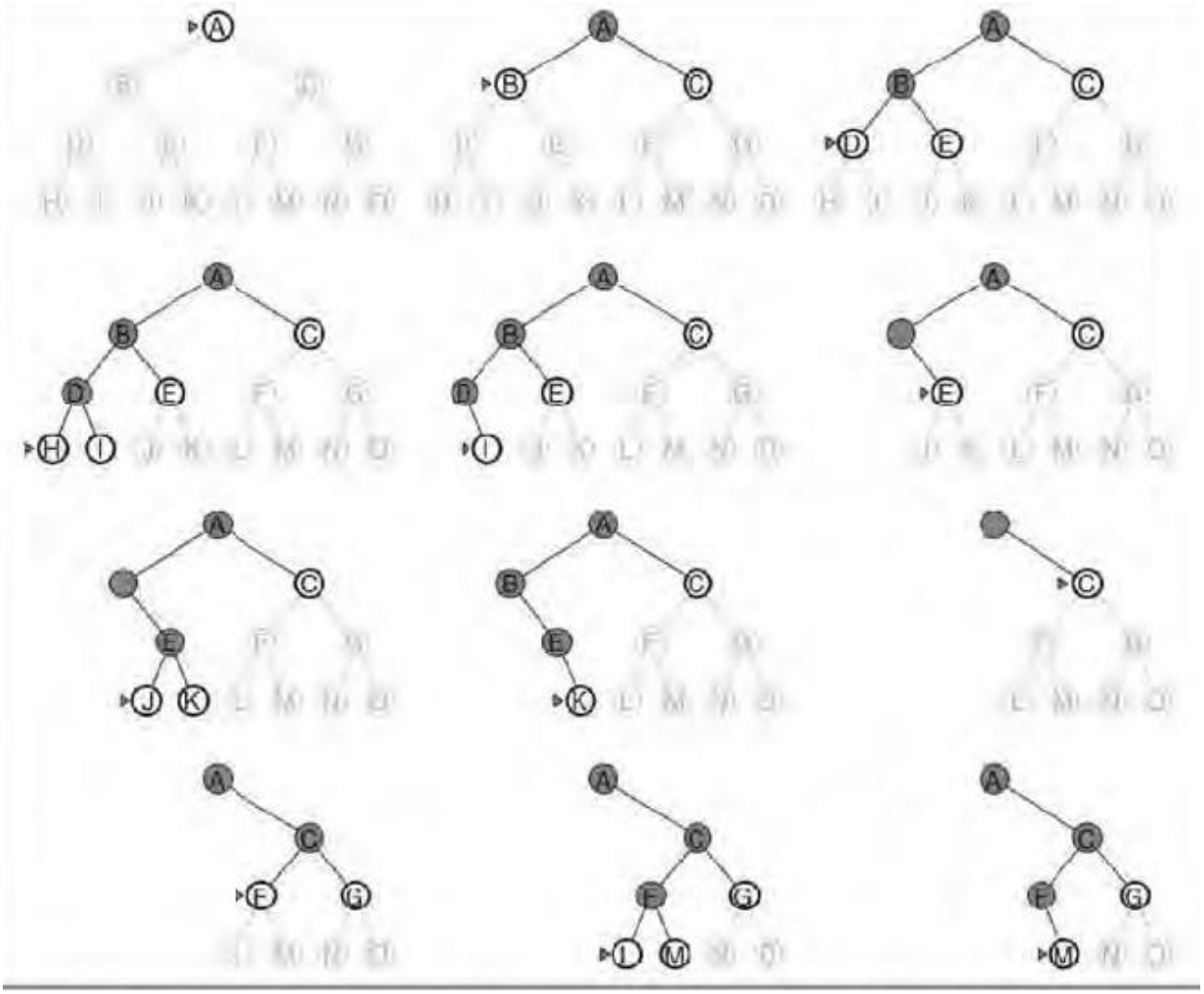
Sin embargo, dicha estrategia de búsqueda no es siempre la más adecuada debido a su complejidad temporal y espacial. Si el factor de ramificación en cada expansión es *b,* en el primer nivel se generan b nodos, por cada uno de ellos en el siguiente nivel se generan b nodos más, por lo que se llegan a b^2 nodos en el segundo, análogamente b^3 nodos en el tercero, y así sucesivamente. Si se llega al estado objetivo en el nivel d, el máximo número de nodos expandidos antes de encontrar una solución es:

$1 + b + b^2 + b^3 + ... + b^d = b^{d+1} - 1 = b^* b^d + 1 = O(b^d)$

La complejidad espacial es la misma que la complejidad temporal porque todos los nodos hojas deben mantenerse en memoria al mismo tiempo.

Búsqueda en profundidad

La búsqueda primero en profundidad (Depth First Search en inglés) siempre expande uno de los nodos en el nivel más profundo del árbol. Sólo cuando la búsqueda llega a un callejón sin salida (un estado no final con cero expansiones) regresa a un nivel superior a buscar por otra vía. En la fig se muestra cómo se ejecuta dicha estrategia de búsqueda:



La búsqueda primero en profundidad tiene muy modestos requerimientos de memoria. Solo necesita almacenar un camino desde la raíz hasta un nodo, junto con los correspondientes nodos hermanos no expandidos en el camino hasta el nodo actual, tal como se aprecia en la figura anterior: una vez que un nodo ha sido expandido, puede ser eliminado de memoria inmediatamente que todos sus descendientes hayan sido explorados.

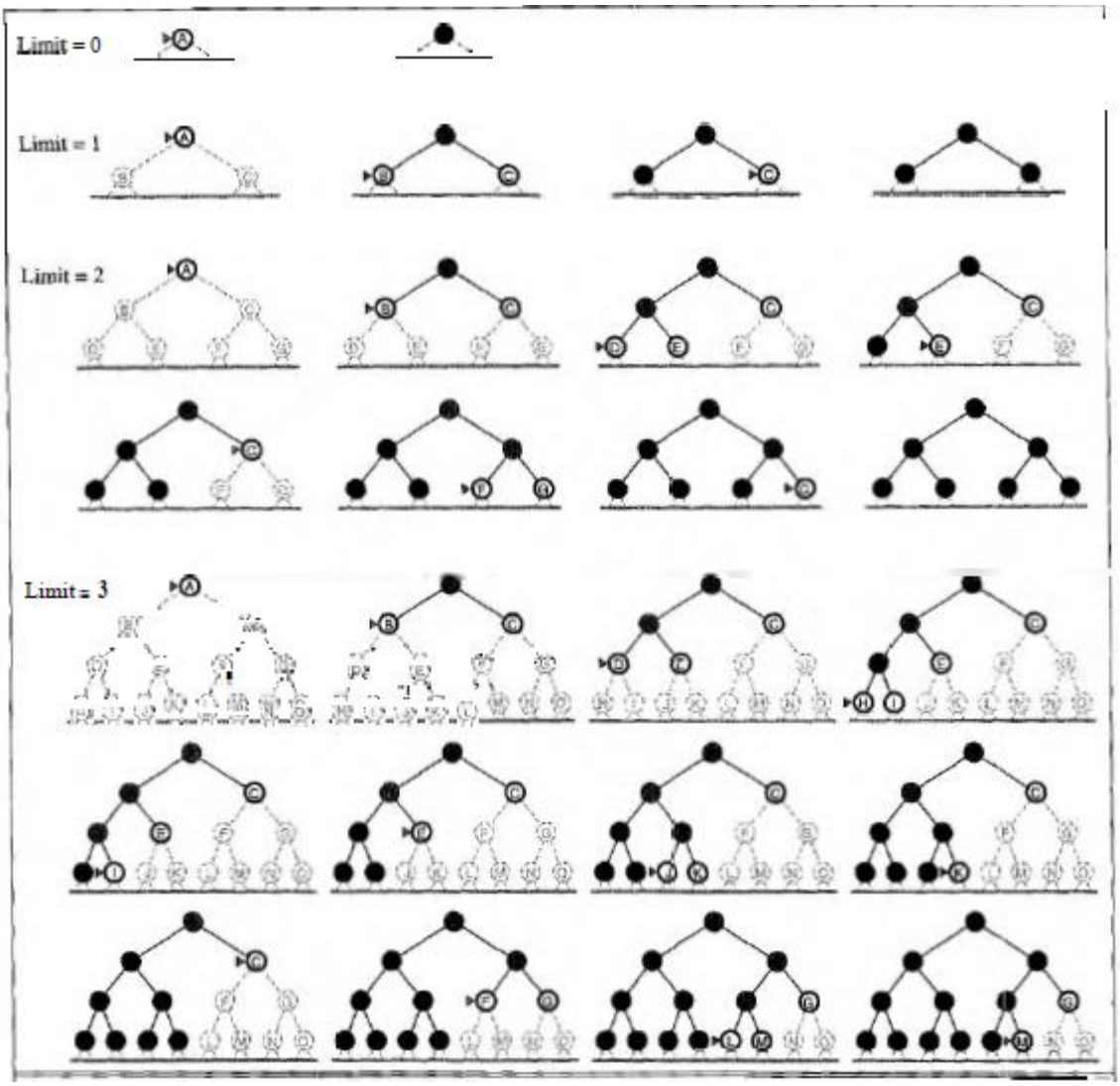
Para un espacio de estados con factor de ramificación *b* y profundidad máxima m, la búsqueda en profundidad requiere almacenamiento solo para b*m +1 nodos (contando el nodo raíz), por tanto es O(b*m). Una variante de dicha estrategia llamada *búsqueda con backtracking* utiliza aún menos memoria: en ella solo un sucesor es generado en cada momento en vez de todos los sucesores, y cada nodo parcialmente expandido recuerda qué sucesor generar después. Esto permite que se requiera almacenamiento solo para m+1 nodos, y efectivamente es la estrategia que se utiliza en las implementaciones de Prolog.

El inconveniente de la búsqueda primero en profundidad es que puede hacer una decisión errónea y enfrascarse en un camino muy largo o incluso infinito cuando otro nodo seleccionado para expandir encontraría la solución mucho más rápido. Luego esta estrategia no es ni completa ni óptima. Su complejidad temporal es $O(b^m)$, (m: máxima profundidad del árbol), puesto que en su peor caso tendría que visitar todos los nodos del árbol de búsqueda para encontrar la solución.

Para lidiar con la deficiencia de no completitud de dicha estrategia de búsqueda, aparece la variante de *Búsqueda Acotada en Profundidad* (Depth Limited Search en inglés), la cual incluye la restricción de que los nodos a un nivel *l* predeterminado sean tratados como si no tuvieran sucesores. Esta variante resuelve el problema de las ramas infinitas. Sin embargo, introduce una nuevo factor de no completitud si l < d, lo que puede pasar perfectamente cuando d es desconocido. También resulta ser no óptimo si l > d, pues en la rama que explore primero puede encontrar una solución que no necesariamente esté más cercana a la raíz del árbol de estados. Su complejidad temporal es $O(b^l)$ y su complejidad espacial es O(b*l).

Búsqueda por profundización iterativa.

La búsqueda por profundización iterativa (Iterative Deepening Search en inglés) es una estrategia general que utiliza la búsqueda acotada en profundidad incrementando en cada iteración el límite l (inicialmente 0, después 1, 2 ,3, ...) hasta que se encuentra un estado objetivo, lo cual ocurre cuando l = d. La profundización iterativa combina las ventajas de la búsqueda primero en profundidad con la búsqueda primero a lo ancho. Como la búsqueda primero en profundidad, su complejidad espacial es O(b*d). Como la búsqueda primero a lo ancho, es completa cuando su factor de ramificación es finito, y además es óptima. En la siguiente figura se pueden ver 4 iteraciones del algoritmo en un árbol con b = 2.



Podría parecer que la búsqueda por profundización iterativa no es eficiente porque los estados son generados múltiples veces (por ejemplo los hijos del nodo raíz son generados d veces). Sin embargo, la mayoría de los nodos están en el último nivel, luego no importa mucho que los niveles superiores del árbol sean generados múltiples veces. En una profundización iterativa, los nodos del nivel d son generados una vez, los del nivel d-1 lo son 2 veces, y así sucesivamente hasta los descendientes directos de la raíz, que son generadosd veces. Por lo tanto el número total de nodos generados es:

$N(BPI) = (d)\ b + (d - l)b^2 + \ldots + (1)\ b^d$

lo que da una complejidad temporal de $O(b^d)$. Podemos compararlo con los nodos generados por la búsqueda primero a lo ancho:

$N(BPA) = b + b^2 + \ldots + b^d + (b^{d+1} - b)$

La diferencia estriba en que la búsqueda primero a lo ancho genera también nodos del nivel d+1, mientras que la profundización iterativa no lo hace, lo que permite que esta última sea en realidad más rápida que la búsqueda primero a lo ancho, a pesar de la repetida generación de estados. **En general, la búsqueda por profundización iterativa es la estrategia de búsqueda no informada**

preferible cuando hay un amplio espacio de búsqueda y la profundidad de la solución es desconocida.

Conclusiones.

Los procesos inferenciales tienen gran repercusión en los sistemas basados en el conocimiento.
Existen diferentes métodos de búsqueda, cada uno, con su particularidad, puede ser aplicado en los sistemas basados en el conocimiento.
Los procesos inferenciales no solo son aplicables a la Inteligencia Artificial, sino a cuestiones prácticas de la vida.

Estudio Independiente.

1. Muestra como quedaría la aplicación de métodos de solución de problemas en el problema de las 8 reinas (ubicar 8 reinas en un tablero de 8*8 sin que se amenacen entre sí).