Sistemas de producción y búsqueda de soluciones

Técnicas de búsqueda

- **Resolución de problemas en Inteligencia Artificial.**
  - En general, podemos afirmar que un problema consiste en:
    - Una descripción de la situación de la que se parte;
    - Una descripción de la situación a la que se quiere llegar;
    - Una descripción de los medios de que disponemos para alcanzar nuestro objetivo.

Técnicas de búsqueda

- En el contexto de la Informática, a partir de un problema, se intenta construir un sistema que lo resuelva.
- Resumidamente, las acciones para construir un sistema que resuelva un problema serían:
  - Definir el problema con precisión (especificación del problema), habitualmente a partir de un enunciado del problema expresado en lenguaje natural:
  - de qué se parte;
  - cuál es el objetivo.
  - Analizar el problema: información para elegir las técnicas.
  - Aislar y representar el conocimiento necesario para resolver el problema.
  - Elegir la mejor técnica que resuelva el problema y aplicarla al problema particular.

Técnicas de búsqueda

- **Si particularizamos la definición de problema dada en el apartado anterior del siguiente modo:**
  - **Entrada: una descripción del estado inicial del mundo.**
  - **Salida: una descripción (parcial) del estado del mundo deseado.**
  - **Medios: una descripción de acciones que puedan transformar un estado del mundo en otro, acciones que son consideradas como operadores o reglas.**

Tendremos el marco conceptual que permite resolver el problema por medio de un sistema de producción.

Ejemplos de resolución de problemas

- **El 8-Puzzle**
  - Entrada: un tablero 3x3 con fichas y dígitos Fig.1.
  - Salida: ordenarlo según una disposición, por ejemplo como en la fig. 2.
  - Medios: si una casilla del tablero está al lado de la casilla vacía se puede desplazar sobre ella.

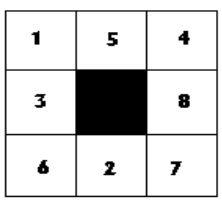


Fig 1

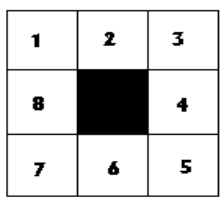


Fig 2

Ejemplos de resolución de problemas

- **El problema de las garrafas de vino.**
  - **Entrada: dos garrafas vacías, una de 4 litros y otra de 3.**
  - **Salida: la garrafa de 4 litros contiene exactamente 2 litros de vino**
  - **Medios: se dispone de un depósito con mucho vino y las únicas operaciones permitidas son llenar cada garrafa en el depósito, vaciarlas y pasar contenido de una a otra hasta que la primera se vacíe o la segunda se llene.**

Sistemas de producción

- **Un sistema de producción consiste en:**
  - Una base de datos/hechos/conocimiento con información sobre el problema;
  - Un conjunto de reglas (operadores);
  - Una estrategia de control;
  - Un aplicador de reglas: ciclo de reconocimiento-actuación.

Espacios de representación

- **Espacio del representación : entorno en el cual se desarrolla el proceso de búsqueda. También se le llama espacio de problema.**
- **Espacio de estados: conjunto de estados que podrían obtenerse si se aplicaran todos los operadores posibles a todos los estados que se fuesen generando. Cada estado representa una situación que se debe considerar como candidata a pertenecer a la solución del problema.**
- **Estado meta: es el estado solución, y debe existir al menos uno.**

Espacios de representación

- La **estrategia de control EC** (también llamada, en el contexto de los sistemas expertos, mecanismo o motor de inferencia). Es la parte más importante de los sistemas de producción y la que establece el proceso de búsqueda de la solución mediante la aplicación de operadores a los distintos estados.
- Con esta perspectiva, la EC se encarga de decidir que estados serán expandidos y decidir qué operador aplicar en cada momento.

Sistemas de producción

- Preguntas que se deben hacer sobre los operadores:
  - ¿se puede aplicar?
  - ¿produce algún cambio en el estado?
  - ¿qué estado elegir para aplicar los operadores?
  - ¿qué sucede si hay varios operadores posibles a aplicar? (resolución de conflictos)
  - Los operadores, que admiten una interpretación como reglas, son acciones simples que pueden transformar un estado en otro. Generalmente se distinguen en los operadores una parte izquierda (el patrón o condición), que determina la aplicabilidad de la regla (se dice también que si se verifica la condición la regla queda sensibilizada), y una parte derecha que describe la operación o acción a llevar a cabo al aplicar el operador (o disparar la regla).

Representación de problemas

- **La representación se da en tres niveles:**
  - **Conceptual: en lenguaje natural**
  - **Lógico- Físico : descripción de estados y operadores, e implementación de las estructuras de datos necesarias para representarlos en un lenguaje (por ejemplo LISP)**

  y en cada uno de ellos deberemos elegir representaciones para dos cosas
  - **Estados**
  - **Operadores**

Representación de problemas

- En un nivel conceptual el estado debe describir
  - todo lo que es necesario para resolver el problema
  - nada que no sea necesario para resolver el problema
  - 8-puzzle:
    - Estados: Localización de cada casilla y del hueco.
    - Operadores: movimiento de las fichas.
- En un nivel lógico debemos representar los estados
  - Estados:
    - matriz 3x3
    - vector de 9 elementos
  - Dependerá de las estructuras permitidas en el lenguaje de programación que usemos

Representación de problemas

- y los operadores (que deben ser lo más generales posibles)
  - 4x9! operadores para pasar de un estado a cualquiera de sus 4 estados sucesores
  - 4x8 operadores que mueven cualquier ficha arriba, abajo, derecha o izquierda.
  - 4 operadores que mueven el hueco arriba, abajo, derecha o izquierda.
    - Evidentemente la última representación es la más adecuada.

Representación de problemas

- **Para el problema de las garrafas de vino**
  - **Nivel conceptual:**
    - **Estados: el único dato relevante es la cantidad de vino que contienen las garrafas, porque lo que contenga la grande no influye en el problema.**
    - **Operadores:**
      - **llena-cuatro: llenar, cogiendo del depósito, la garrafa de 4 l.**
      - **llena-tres: llenar, cogiendo del depósito, la garrafa de 3 l.**
      - **vacía-cuatro: vaciar en el depósito la garrafa de 4 l.**
      - **vacía-tres: vaciar en el depósito la garrafa de 3 l.**
      - **echa-la-cuatro-a-la-tres**
      - **echa-la-tres-a-la-cuatro**

Representación de problemas

- **Nivel lógico-físico:**
  - Un estado va a ser identificado como un par de números (x,y) donde cada representa el número de litros que contiene la garrafa de cuatro litros, e y la de tres.
  - Los operadores se pueden implementar como una regla:

    llena-cuatro(x,y):

    precondición: x<4

    acción: construir el estado (4,y)
  - Implementación de las estructuras de datos en LISP.

Grafos

- Un **grafo** viene dado por un conjunto de **nodos** y un conjunto de **arcos** entre los nodos.
- Grafo **dirigido**: grafo en el que (A,B) se considera diferente de (B,A); ejemplo red bayesiana.
- En caso contrario es **no dirigido.**
- **Camino**: sucesión de nodos tales que entre dos de ellos consecutivos existe un arco.
- **Grafo conexo:** entre dos nodos cualesquiera existe un camino.

Grafos

- En un grafo no dirigido cualquier camino cerrado recibe el nombre de **ciclo**. Si no contiene ciclos se denomina **árbol** (no dirigido).
- Si el grafo es dirigido y tenemos un camino cerrado, si podemos recorrerlo siguiendo el sentido de los arcos y volvemos al punto de partida tenemos un **ciclo**. Si no tenemos un **bucle**.
- Los grafos dirigidos **acíclicos** (GDA) son aquellos que no contienen ciclos.
- En grafos acíclicos hablaremos de nodos **padre** e **hijo**, en vez de antecesores y sucesores.

Grafos en búsqueda

- Los GDA conexos que no contienen bucles se denominan **poliárboles.**
- **Nodos:** elementos en el espacio de estados que representan situaciones válidas en el dominio. Es decir, nodo y estados son sinónimos.
- **Expansión** de un nodo: obtener todos sus posibles sucesores en el grafo de búsqueda a través de la aplicación de todos los operadores.

Grafos en búsqueda

- **Conceptos**
  - **Nodo o estado cerrado**: aquel que ya ha sido expandido.
  - **Nodo o estado abierto**: aquél en el que todavía queda por aplicar algún operador.
  - **Coste de un arco**: es un valor numérico que refleja el tiempo requerido para aplicar un operador a un estado en el proceso de búsqueda.
  - **Factor de ramificación**: número medio de descendientes de un nodo, o lo que es lo mismo, el número medio de operadores aplicados a dicho camino.
  - **Profundidad**: longitud del camino más corto desde el estado inicial a una meta. O la longitud de secuencia más corta de operadores que resuelven el problema.

Resumen

- **Los procesos de búsqueda tienen sentido en problemas que reúnen una serie de características:**
  - **Cabe la posibilidad de asociar un conjunto de *estados* a las diferentes situaciones en que se puede encontrar el objeto del dominio sobre el que se define el problema.**
  - **Hay una serie de *estados iniciales* desde los que empezar el proceso de búsqueda.**
  - **Existen ciertos *operadores*, tal que un operador aplicado sobre un estado producirá otro estado.**
  - **Existe al menos un *estado meta* o estado *solución***

Reglas de producción

- **Taxonomías de los sistemas de producción/búsqueda/estrategia de control:**
  - **Hacia adelante / hacia atrás / bidireccional**
  - **Irrevocable / tentativa**
  - **Informada / no informada**

Búsqueda sin información del dominio

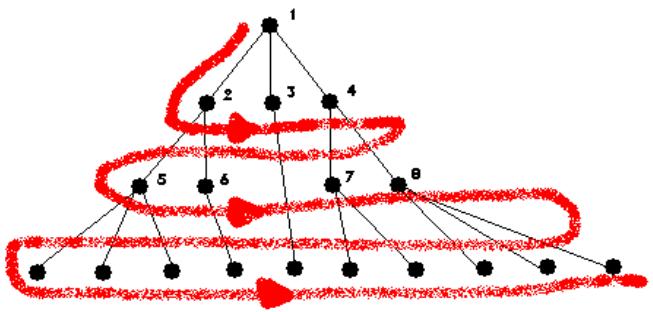
- También se denominan técnicas de **búsqueda ciega**, porque realizan una búsqueda sistemática y objetiva (en el sentido de que el control del proceso no depende del problema concreto que se esté resolviendo)
- Por el contrario las técnicas de **búsqueda heurística** realizan una búsqueda informada e intentan optimizar dicho proceso eligiendo los caminos que a priori van a suponer un menor coste.

Búsqueda sin información del dominio

- **Objetivos:**
  - Encontrar el camino óptimo entre la descripción del problema o estado inicial y el estado meta.
  - A veces basta con devolver el estado meta y no es necesario conocer todo el camino.
- **Características:**
  - No dejar (a priori) ningún nodo sin explorar.
  - No explorar un nodo más de una vez.

Búsqueda en amplitud

- Es aquél procedimiento de control en el que se revisan todas las trayectorias de una determinada longitud antes de crear una trayectoria más larga.
- Es decir, no se genera ningún nodo de nivel N hasta que no se hayan obtenido todos los del nivel N-1.



Búsqueda en amplitud

- **Algoritmo de búsqueda en amplitud**
  - 1. Crear una lista de nodos llamada ABIERTA e "inicializarla" con un único nodo raíz, al que se le asigna el estado inicial del problema.
  - 2. Hasta que ABIERTA esté vacía o se encuentre una meta realizar las siguientes acciones:
    - 2.1 Extraer el primer nodo de ABIERTA y llamar a ese nodo *m.*
    - *2.2* Expandir *m* (generar todos los sucesores). Para cada operador aplicable y cada forma de aplicación:
      - 1) Aplicar el operador a m, obtener un nuevo estado y crear un puntero que permita saber que su predecesor es *m*.
      - 2) Si el nuevo estado generado es meta, salir del proceso iterativo iniciado en 2.2 y devolver dicho estado.
      - 3) Incluir el nuevo estado al final de ABIERTA (una vez completado este proceso para todos los sucesores de *m* - cuando no se haya encontrado una meta- se continúa el proceso iterativo en el paso 2)

Búsqueda en amplitud

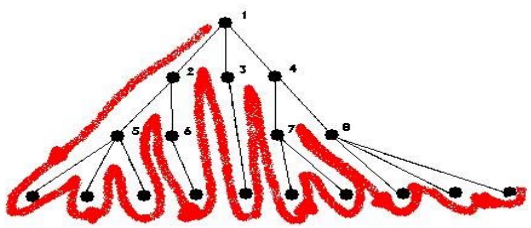
- En este algoritmo la lista ABIERTA va a funcionar como una cola FIFO.
- Los nodos que haya en ABIERTA serán aquellos que hayan sido generados, pero que todavía no han sido expandidos.
- Los elementos que van a ser expandidos se toman al comienzo de la lista ABIERTA.
- Sus sucesores se añaden al final.
- De esta forma siempre se expandirán los nodos más antiguos.

Búsqueda en amplitud

- **Ventajas: si el problema tiene una solución este procedimiento garantiza el encontrarla. Si hubiera varias soluciones se obtiene la de menor coste (la óptima), es decir, la que requiere un menor número de pasos (si consideramos un coste uniforme de aplicación de los operadores)**
- **Desventajas: si el nivel de profundidad asociado a la solución es significativamente menor que el factor de ramificación se expandirían demasiados nodos inútilmente. Por otro lado la principal desventaja_de este método es el espacio de almacenamiento requerido. Esto lo hace prácticamente inviable para problemas complejos, como suelen ser los del mundo real.**

Búsqueda en profundidad

- Es aquél procedimiento de control en el que se centra en expandir un único camino desde la raíz.
- En el caso de llegar a un *callejón sin salida* se retrocede hasta el nodo más cercano desde donde se puede tomar una ruta alternativa para poder seguir avanzando.
- Para llevar a cabo este tipo de búsqueda debe utilizarse una estructura de tipo pila (LIFO) que vaya almacenando los nodos generados.
- Suele establecerse el llamado *límite de exploración*, que marca la máxima longitud que puede alcanzar cualquier camino desde la raíz durante el proceso de búsqueda.



Búsqueda en profundidad

- **Algoritmo de búsqueda en profundidad**
  - 1. Crear una lista de nodos llamada ABIERTA e "inicializarla" con un único nodo raíz, al que se le asigna el estado inicial del problema.
  - 2. Hasta que ABIERTA esté vacía o se encuentre una meta realizar las siguientes acciones:
    - 2.1 Extraer el primer nodo de ABIERTA y llamar a ese nodo *m.*
    - 2.2 Si la profundidad de m es igual a lp regresar a 2.1. En caso contrario continuar.
    - 2.3 Expandir m (generar todos los sucesores). Para cada operador aplicable y cada forma de aplicación:
      - 1) Aplicar el operador a m, obtener un nuevo estado y crear un puntero que permita saber que su predecesor es *m*.
      - 2) Si el nuevo estado generado es meta, salir del proceso iterativo iniciado en 2.1 y devolver dicho estado.
      - 3) Incluir el nuevo estado al principio de ABIERTA en un orden arbitrario.
        - Si algún sucesor de *m* es meta, abandonar el proceso iterativo señalado en 2.1devolviendo el camino de la solución que se obtiene recorriendo los punteros de sus antepasados.
        - Si algún sucesor de *m* se encuentra en un callejón sin salida eliminarlo de ABIERTA y continuar en 2.2

Búsqueda en profundidad

- Ventajas: la principal ventaja de esta algoritmo radica en el reducido valor de su complejidad espacial. Cuando existen múltiples soluciones posibles la eficiencia del algoritmo aumenta.
- Desventajas: la dificultad estriba en el tiempo requerido. El algoritmo puede dedicarse a recorrer un camino demasiado largo que no conduzca a ninguna solución. Es más, si no se guarda constancia de los nodos que forman el camino recorrido se podría caer en ciclos y el proceso no acabaría.
- El problema por tanto es determinar cuál debe ser *lp.* Si éste es inferior a la longitud real del camino de la solución, ésta nunca se encontraría, y si es mucho mayor sería ineficiente. Esta es la razón por la que *lp* debería llamarse *límite de exploración*.

Búsqueda con retroceso

- Es un método que a diferencia de los algoritmos en amplitud y en profundidad (que consideran todos los sucesores) solamente expande un sucesor en cada iteración, restringiendo por lo tanto el espacio de estados considerado.
- Cuanto mejor sea el criterio para limitar el número de estados considerados más eficiente será el proceso de búsqueda.
- Es decir, es como el recorrido en profundidad pero nos ahorramos tener que expandir todos los nodos para obtener sus sucesores. El camino sigue avanzando por ese sucesor, y si nos encontramos con un callejón sin salido retrocedemos hasta el primer antepasado desde el que todavía partan caminos inexplorados.

Búsqueda con retroceso

- **Algoritmo de búsqueda en profundidad**
  - **1. Crear una lista de nodos llamada ABIERTA e "inicializarla" con un único nodo raíz, al que se le asigna el estado inicial del problema.**
  - **2. Hasta que ABIERTA esté vacía o se encuentre una meta o se devuelva fallo realizar las siguientes acciones:**
    - **2.1 Si ABIERTA está vacía terminar con fallo: en caso contrario continuar.**
    - **2.2 Extraer el primer nodo de ABIERTA y llamar a ese nodo *m.***
    - **2.3 Si la profundidad de m es igual a lp o si *m* no tiene más sucesores posibles (que no hayan sido examinados anteriormente) eliminarlo de ABIERTA y regresar a regresar a 2. En caso contrario continuar.**
    - **2.4 Generar un "nuevo" sucesor *m´* de *m.* e introducirlo al principio de ABIERTA. , creando un puntero a m, y señalar que dicha rama ya ha sido considerada.**
      - **4.1) Si *m´* es meta, abandonar el proceso iterativo iniciado en el paso 2 devolviendo el camino de la solución, que se obtiene recorriendo los punteros de sus antepasados.**
      - **4.2) Si *m´* se encuentra en un *callejón sin salida* eliminarlo de ABIERTA. Se continúa el proceso iterativo en el paso 2.**

Búsqueda con retroceso

- Ventajas: la principal ventaja de esta algoritmo respecto al de profundidad es la de necesitar un menor espacio de almacenamiento. Sólo hay que recordar en cada instante un sucesor del nodo seleccionado. Otra ventaja es que no se generan las ramas del árbol que se encuentran después (a la derecha) de la solución.
- Desventajas: Vuelve a ser el problema determinar cuál debe ser *lp.* Para conocer su valor deberíamos saber de antemano en qué nivel se encuantra la solución

Algoritmo general de búsqueda en grafos

1) ABIERTOS := (nodo_inicial); RESUELTO:=FALSO;
2) **mientras que** ABIERTOS no es vacía **Y NO** RESUELTO
3) **N:=quitar elementos de ABIERTOS;**
   E:=estado asociado a N
4) si **E es un estado objetivo**
5) entonces **RESUELTO:=verdad**
   si no para cada operador O **hacer**
6) **si** O se puede aplicar a E
7) **entonces** crear un nodo correspondiente
   al estado obtenido por aplicación de O a E
   y añadir ese nodo a ABIERTOS

si RESUELTO

8) **entonces** devuelve el estado objetivo (y si se requiere una explicación, el camino por el que hemos llegado a él)
9) **si no** informa de que el objetivo no puede ser alcanzado

Algoritmo general de búsqueda en grafos

- En los pasos 6 y 7 se encuentra implicada la parte del sistema de producción que dispara las reglas.
- El tipo de búsqueda depende del paso número 7:
  - Si el algoritmo no sabe nada del problema que está resolviendo se llama búsqueda sin información del dominio.
  - Los nodos del grafo son generados a partir del paso 7.
    - Si los nuevos nodos son añadidos al final de la lista ABIERTOS la búsqueda se llama búsqueda en amplitud. Así se consigue que los nodos en abierto estén ordenados según su profundidad, en orden decreciente: los menos profundos al principio y los más profundos al final. La lista ABIERTOS pasa a ser una *cola FIFO.*
    - ABIERTOS sería una lista de todos los nodos que se han generado pero que todavía no han sido expandidos

Herramientas de IA

- **Intérpretes generales de sistemas de producción:**
  - Lenguajes basados en reglas:
    - OPS5 (Brownston, en 1985, de CMU, en Common Lisp usa RETE);
    - CLIPS (C Language Integrated Production System) (de NASA, permite la integración con lenguajes como C y Ada).
    - Lenguajes basados en lógica: el más extendido es PROLOG.
  - Armazones de sistemas expertos:
    - EMYCIN;
    - en entornos de IA (KEE, KnowledgeCraft, LOOPS, …).
  - Arquitecturas generales de resolución de problemas:
    - GPS (Newell, Shaw, Simon) 1963;
    - SOAR (Newell, Laird, …) 1987.