Tema 3. Búsqueda en espacios de estados.

Diseño de un agente deliberativo.

El agente dispone de un modelo del mundo en el que habita. Dispone de un modelo de los efectos de sus acciones sobre el mundo, y es capaz de razonar sobre esos modelos para decidir que hacer para conseguir un objetivo.

La búsqueda en un espacio de estados.

El espacio de estados es la representación del conocimiento a través de las acciones del agente. La búsquedda en el espacio de estados es la resolución del problema mediante proyección de las distintas acciones.

Descripción de un problema.

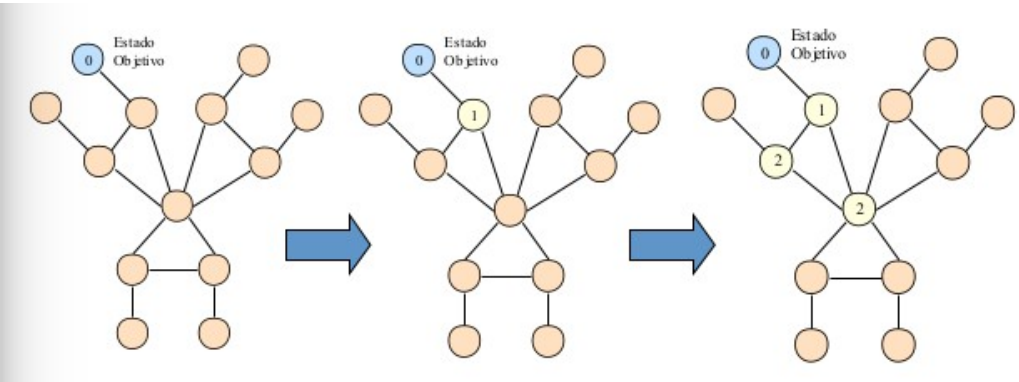
Estados, estado inicial, acciones, objetivo, costo de las acciones.

El mundo de bloques.

Una estructura de grafo dirigido puede ser útil para buscar secuencias de acciones que nos lleven al objetivo final. Un nodo representa un estado del sistema y un arco una posible acción. La acción aplicada al estado que representa al nodo origen, producirá el estado del nodo destino. Se denomina **grado de estados.**

A la secuencia de acciones que lleva al agente desde un estado inicial hasta un estado destino se denomina plan. La búsqueda de dicha secuencia se denomina planificación (grafos explícitos o implícitos).

Búsqueda.



CURSOS DE INGLÉS EN EL EXTRANJERO

La inversión más inteligente para tu futuro

- 80 años de experiencia en educación internacional.
- 97% de recomendación entre nuestros estudiantes.
- 40 escuelas de inglés acreditadas: Reino Unido, Irlanda, Estados Unidos, Canadá, Australia y Nueva Zelanda.
- Cursos para todos los niveles y objetivos: inglés general, de negocios, preparación de exámenes, larga duración, entre otros.

DESCARGA **EL CATÁLOGO** GRATUITO

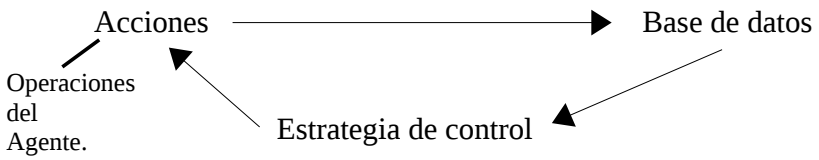
KAPLANINTERNATIONAL.COM/ES

KAPLAN INTERNATIONAL ENGLISH

Ejemplos.

→ Problema del mono y los plátanos: un mono en la puerta de una habitación. En el centro hay un plátano colgado del techo. En la ventana hay una caja, que el mono puede mover y a la que puede encaramarse. El mono puede realizar las siguientes acciones: desplazarse al centro, del centro a la ventana, empujar la caja, subirse/bajarse de la caja, coger el plátano. El problema es encontrar la secuencia que permita al mono coger el plátano.
→ Ejemplo del mapa de carreteras.
→ Ejemplo del problema de la aspiradora: suponemos que el agente conoce su entorno y sus acciones. Si meto incertidumbre en lo que conozco sobr eel estado, también puedo meter incertidumbre en lo que hacen las operaciones (su efecto).

Sistemas de búsqueda y estrategias.



Procedimiento búsqueda.

1. DATOS ← base de datos inicial.
2. until Datos satisface la condición de terminación do
3. begin
4. select alguna acción A en el conjunto de acciones que pueda ser aplicada a DATOS.
5. DATOS ← resultado de aplicar A a DATOS.
6. end

Estrategias de control.

→ Estrategias irrevocables: no guardo nada.
→ Estrategias tentativa:
- · Retroactivas: guardo una secuencia única de estados. Un camino único de base de datos.
- · Búsqueda en grafos: guardo todas las bases de datos que se usan.

Estrategias irrevocables.

Se usan en problema en el que el orden en el que se aplican los operadores no afecta a que puedas encontrar la solución.
Hay problemas en los que tendrías que usaar información de antes, por lo que no podríamos resolver ese problema.
En cada momento, el grafo explícito lo constituye un único nodo, que incluye la descripción completa del sistema en ese momento:
→ Se selecciona una acción A. (Solo tengo un estado en cada momento)
→ Se aplica sobre el estado del sistema E, para obtener un nuevo estado E'.
→ Se borra de memoria E y se sustituye por E'.

Estrategia retroactiva (Backtracking)

Guarda información, pero restringe la memoria para guardar lo mínimo indispensable para devolver el camino.
Hay una secuencia única de base de datos que es la que se recupera al final.
En memoria sólo guardamos un hijo de cada estado; se mantiene el camino desde el estado inicial hasta el actual.
El grafo explícito es realmente una lista.
El proceso para cuando hemos llegado al objetivo y no deseamos encontrar más soluciones, o bien no hay más operadores aplicables al nodo raíz.
Se produce una vuelta atrás cuando:
→ Se ha encontrado una solución, pero deseamos encontrar una alternativa.
→ Se ha llegado a un límite en el nivel de profundidad explorado o el tiempo de exploración en una misma rama.
→ Se ha generado un estado que ya existía en el camino.
→ No existe reglas aplicables al último nodo de la lista.

Búsqueda en grafos.

En memoria se guardan todos los estados, de forma que la búsqueda puede proseguir por cualquiera de ellos:
1. Seleccionar un estado del grado.
2. Seleccionar un operador aplicable sobre el estado.
3. Aplicar el operador para obtener un nuevo nodo.
4. Añadir el arco del primer estado con el segundo al grafo.
5. Repetir el proceso.

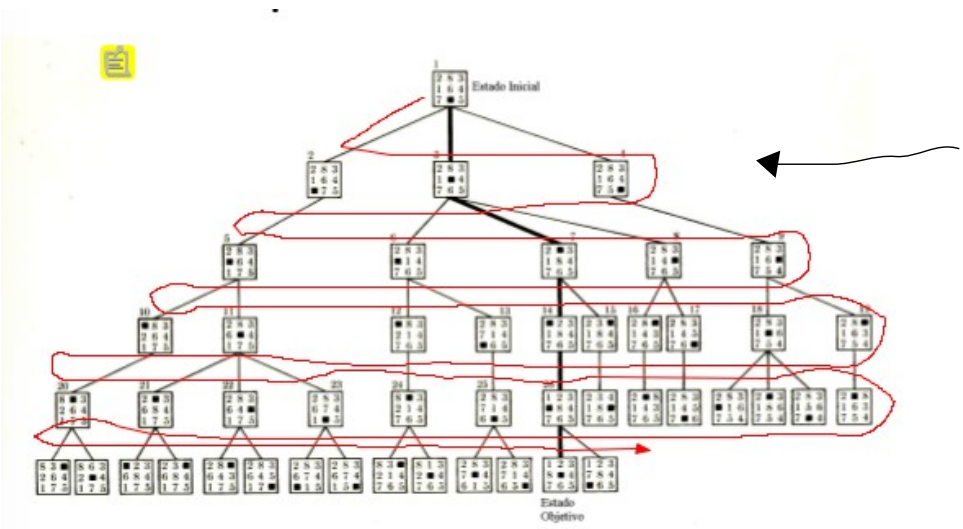
Medidas del comportamiento de un sistema de búsqueda.

→ Completitud: garantía de encontrar a solución si esta existe.
→ Optimalidad: garantía de encontrar la solución óptima.
→ Complejidad en tiempo: ¿cuánto tiempo se requiere para encontrar la solución?
→ Complejidad en espacio: ¿cuánta memoria se requiere para realizar la búsqueda?

Búsqueda sin información.

Búsqueda en anchura.

Utiliza una cola para manejar los nodos.



Mucho gasto en tiempo y en memoria.

Características:
→ Completo: encuentra la solución si existe y el factor de ramificación es finito en cada nodo.
→ Optimalidad: si todos los operadores tienen el mismo coste, encontrará la solución óptima.
→ Eficencia: buena si las metas están cercanas.
→ Problema: consume memoria exponencial.

Búsqueda con costo uniforme.

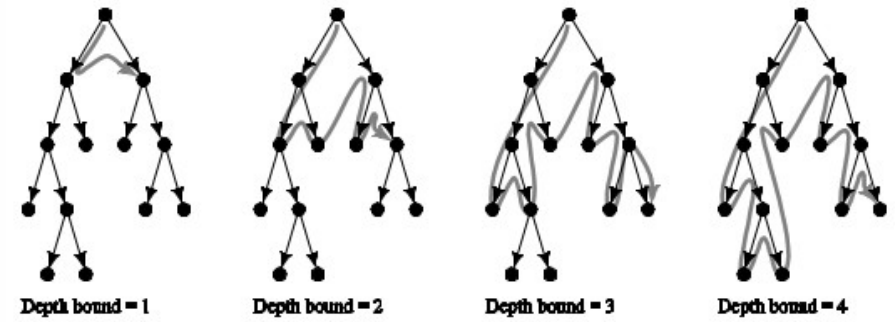
Búsqueda en profundidad.

Igual que la búsqueda en anchura cambiando FIFO por LIFO.

Búsqueda en profundidad retroactiva.

→ Completitud: no asegura encontrar la solución.
→ Optimalidad: no asegura encontrar la solución óptima.
→ Eficiencia: bueno cuando las metas están alejadas del estado inicial, o hay problemas.
→ No es bueno cuando hay ciclos.

Descenso iterativo.



Búsqueda bidireccional.

Búsqueda con información.

Heurísticas.

→ Si se tiene conocimiento perfecto: algoritmo exacto.
→ Si no se tiene conocimiento: búsqueda sin información.
→ En la mayor parte de los problemas que resuelven los humanos, se está en posiciones intermedias.
→ Heurística: conocimeinto parcial sobre un problema que permite resolver problemas eficientemente en ese problema.
→ Un proceso que me sirva de guía sobre lo que tengo que hacer.
→ Entendemos la heurística como una función. Evaluamos el estado del problema con esa función: si f(e1) < f(e2) cogemos e1. Es decir, que e1 es más prometedor que e1. Que cojamos e1 no significca que solo vayamos a ver ese, si no que es el que vamos a mirar primero.
→ Son criterios, métodos o principios para decidir cuál de entre varias acciones promete ser la mejor para alcanzar una determinada meta.

→ Es un método para resolver problemas que en general no garantiza la solución óptima, pero que en media produce resultados satisfactorios en la resolución de un problema.
→ Encapsula el conocimiento específico que se tiene sobre un problema, y sirve de guía para que un algoritmo de búsqueda pueda encontrar una solución válida aceptable.
→ Puede devolver siempre soluciónes óptimas bajo ciertas condiciones.

Ajedrez.

El jugador hace un proceso deliberativo. ¿Qué puedo hacer?

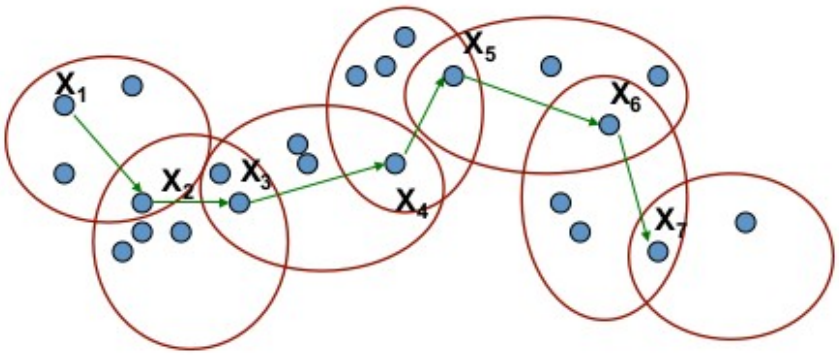
Mapa de carreteras.

8-puzzle.

Tenemos que hacer una función que me determine si el estado en el que estoy me va a llevar de una manera correcta y más sencilla a la solución del problema.
Implementaremos heurísticas como funciones que devuelven un valor numérico, cuya maximización o minimización guiará al proceso de búsqueda a la solución.

Métodos de escalada.

La heurística tiene que ser un esquema lo suficientemente simple y fácil como para reducir el coste de búsqueda.
Si dibujamos las soluciones como puntos en el espacio, una búsqueda local consiste en selecccionar la solución mejor en el vecindario de una solución inicial, e ir viajando por las soluciones del espacio hasta encontrar un óptimo.



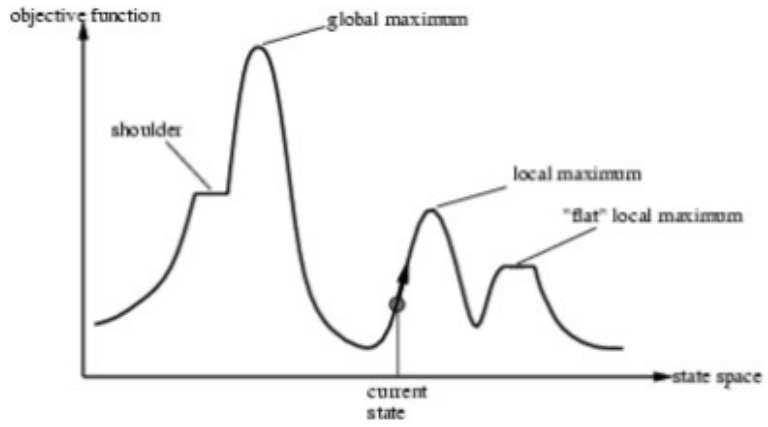
Algoritmo de escalada simple.

Estas en un estado, generas el primer descendiente. Si el primero es peor que el actual, generas otro. Así hasta que encuentras uno mejor, y te mueves a ese. Si no encuentras un descendiente mejor, estás en la solución (aunque no hay garantía).

Algoritmo de escalada por la máxima pendiente.

La escalada simple avanza cuando hay una mejora, y la escalada por la máxima pendiente avanza si encuentra el mejor de todos.
Características:
→La solución del problema depende muchísimo de la función heurística.
→ Completitud: no tiene porque encontrar la solución.
→ Admisibilidad: no siendo completo, aún menos será admisible.
→ Eficiencia: rápido y útil si la función es monótona (de)creciente.
→ Solución completa: solución correcta pero que no sabemos si es óptima.

En los métodos de escalada, los principales problemas son que se encuentra un máximo local, o que se encuentre una meseta.



Algunas varriacoines estocásticas.

→ Algoritmo de escalada estocástico: las variaciones estocásticas son en las que se toma algún tipo de decisión aleatoria.
→ Algoritmo de escalada de primera opción.
→ Algoritmo de escalada de reinicio aleatorio: se generan varias soluciones aleatorias, y se escoge la mejor de ellas. Esto no quiere decir que sea la óptima, si no que es la mejor de las que se han generado.
→ Enfriamiento simulado: complejo y muy usado.

Algoritmo de enfriamiento simulado.

Tiene una explicación en la naturaleza: nos fijamos en un fenómeno de ésta, que está mejorando algo, y lo utilizas en un algoritmo.
→ Es un método de búsqueda local.
→ Se basa en principios de termodinámica.
→ Al contrario que otros métodos de ascensión de colinas, permite visitar soluciones peores que la actual para evitar óprimos locales.

Es mejor a un estado peor de la solución al principio del proceso, no cuando ya estoy cerca de encontrarla.

En el campo de la termodinámica, se simuló el proceso de enfriamiento en sistemas de partículas:

$$P[\delta E] = e^{-\frac{\delta E}{k \cdot T}}$$

δE = diferencia de energía del sistema.
T = temperatura actual del sistema.
k = constante física.

Analogía entre el proceso de enfriamiento y el algoritmo de enfriamiento simulado:
→ Los estados por los que pasa el sistema físico de partículas equivalen a las soluciones factibles del algoritmo.
→ La energía E del estado actual del sistema es el valor de la función objetivo de la solución actual.
→ Un cambio de estado en el sistema equivale a explorar el entorno de una solución y viajar a una solución vecina.
→ El estado final estable es la solución final del algoritmo.
→ La solución inicial se puede generar de forma aleatoria, por conocimiento experto, o por medio de otras técnicas.
→ La actualización de temperatura también es heurística.
→ Número de vecinos a generar: fijo, dependiente de la temperatura, etc.

Tanto la temperatura inicial como la temperatura final son parámetros de entrada al algoritmo. Es difícil asignar un valor concreto a T por lo que la condición de parada se suele sustituir por un número específico de iteraciones a realizar.

Ventajas:
→ Tiene capacidad para salir de óptimos locales.
→ Es eficiente.
→ Es fácil de implementar.
Inconvenientes:
→ Encontrar la temperatura inicial Ti, el método de actualización de temperatura, el número de vecios a generar en cada estado y el númer de iteraciones óptimo es una tarea que requiere de muchas pruebas de ensayo y error hasta que ajustamos los parámetros óptimos.

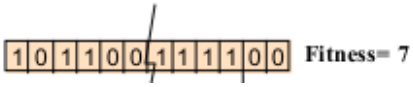
Pese a todo, el algoritmo puede proporcionar soluciones mucho mejores que utilizando algoritmos no probabilísticos.

Algoritmos genéticos.

Se va guardando una secuencia.

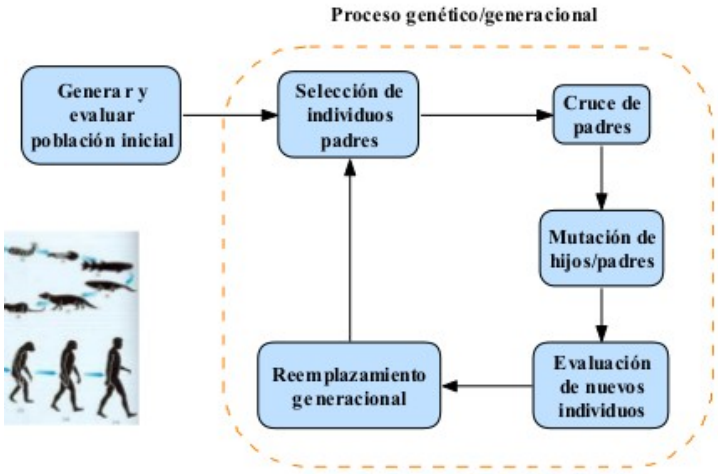
La simulación de procesos naturales es un campo de investigación muy amplio.

Los algoritmos genéticos son algoritmos de optimización basados en el proceso de la evolución natural. En un proceso de evolución, existe una población de individuos. Los más adecuados a su entorno se reproducen y tienen descendencia (a veces con mutaciones). Por tanto, no necesitan partir de un nodo inicial. Su objetivo es encontrar una solución cuyo valor de función objetivo sea **óptimo.**



Valor numérico que dice lo buenos que son los atributos frente al problema. Un mayor valor i indica que el individuo es mejor.

Proceso de un algoritmo genético:

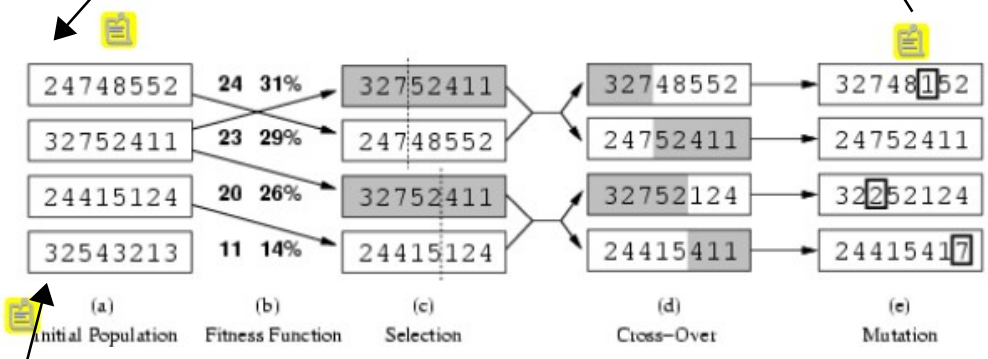


Ejemplo.

Tablero en el que echo 8 reinas al azar.

Cada vector es un tablero.

En la mutación puede llegarse a un estado que no es posible. Hay que controlarlo.



El cuarto es eliminado porque el individuo no es bueno.

Búsqueda primero el mejor.

Algoritmo A*

Función heuristica: f(n) = g(n) + h(n);

ABIERTOS contiene el nodo inicial, CERRADOS está vacío.

Comienza un ciclo que se repite hasta que se encuentra solución o hasta que ABIERTOS queda vacío.

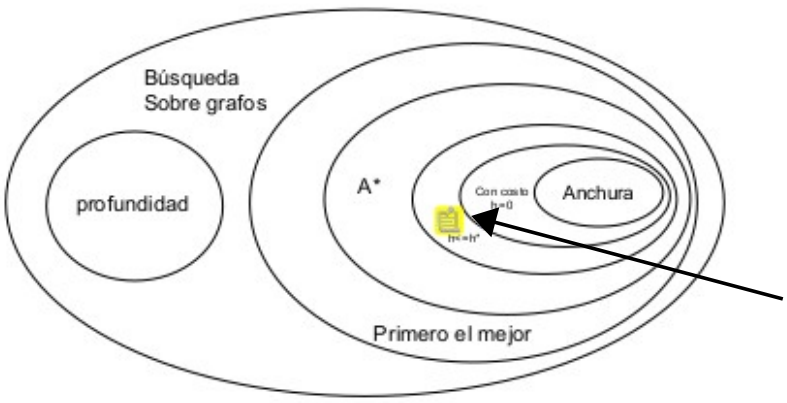
→ Seleccionar el mejor nodo de ABIERTOS.
→ Si es un nodo objetivo terminar.
→ En otro caso se expande dicho nodo.
→ Para cada uno de los nodos sucesores:
- · Si está en ABIERTOS insertarlo manteniendo la información del mejor padre.
- · Si está en CERRADOS insertarlo manteniendo la información del mejor padre y actualizar la información de los descendientes.
- · En otro caso, insertarlo como un nodo nuevo.

→ Características:
f(n) = (1-w) * g(n) + w*h(n) $w \in [0 , 1]$
- · Completitud: si existe solución, la encuentra.
- · Admisibilidad: si hay solución óptima, la encuentra si
  - el número de sucesores es finito para cada nodo.
  - $c(ni, nj) > \delta > 0$ en cada arco, y
  - la función h(n) es admisible: $h(n) \leq h^*(n)$

h (n) = solución más corta en camino recto
h*(n) = solución óptima
n = número de nodos.

(saber si el algoritmo es capaz de devolver la solución óptima)
Es un algoritmo que si tiene garantías.

Algoritmos de búsqueda.



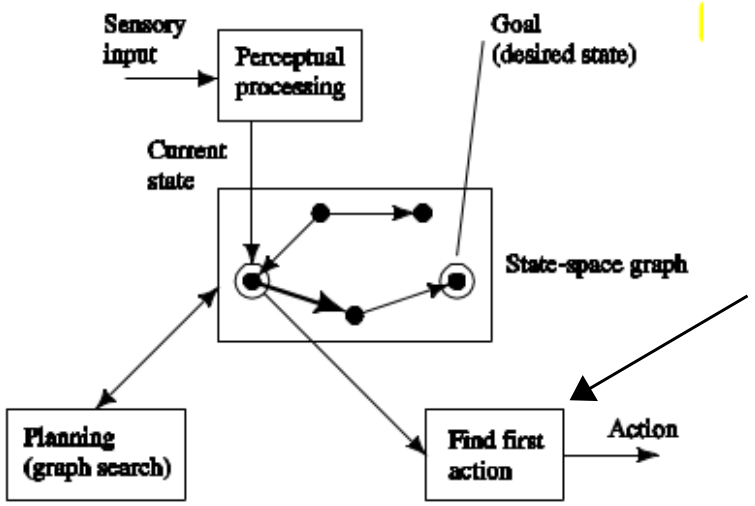
Algoritmos admisibles.

→ Dificultades del proceso:

· Los algoritmos utilizan dos recursos muy importantes: memoria y tiempo.
· Los procesos de percepción no siempre pueden obtener la información necesaria acerca del estado del entorno.
· Las acciones pueden no disponer siempre de modelos de sus efectos.
· Pueden haber otros procesos físicos, u otros agentes, en el mundo.
· En el tiempo que trancurre desde la construcción de un plan, el mundo puede cambiar de tal manera que el plan ya no sea adecuado.
· Podría suceder que se le pidiese al agente acutar antes de que puediese completar una búsquqeda de un estado objetivo.
· Aunque el agente dispuesiera de tiempo suficiente, sus recursos de memoria podrían no permitirle realizar la búsqueda de un estado objetivo.

→ Arquitectura percepción/planificación/actuación.

· Poder dar respuesta a situaciones problemáticas que ya conocemos.
· Ejemplo: aspiradora que está en una habitación. Esa habitación si sabemos si esta limpia o no, pero la de al lado no. Pero realizando una serie de acciones, llegaremos a la siguiente habitación y podremos saber como está.

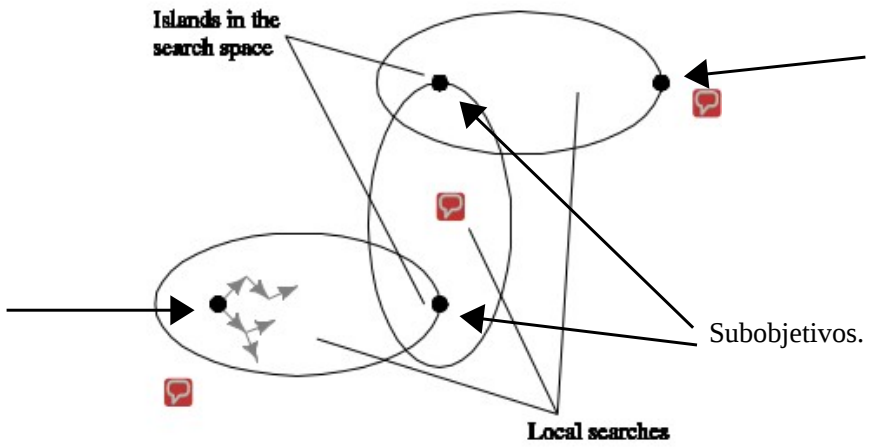


La primera acción es la más segura, ya que donde estamos vamos a conocer más que otro lugar en el que no podemos percibir.

Heurísticas sobre el proceso de búsqueda.

Búsqueda orientada a subobjetivos.

Tengo un problema que puedo solucionar con algún algoritmo que ya conocemos, pero conocemos algunos atributos más. Ejemplo: saber el objetivo, y podemos marcar algunos objetivos intermedios.



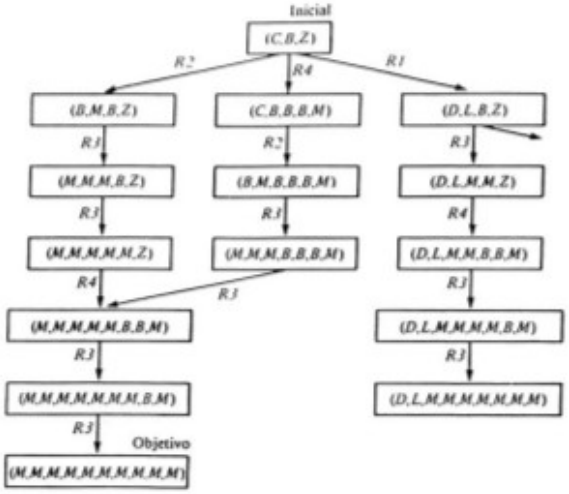
Búsqueda jerárquica.

Una única acción a un nivel muy abstracto corresponde a acciones de bajo nivel.
Ejemplo: un brazo robot quiere coger un objeto. Se puede dividir en problemas más pequeños: mover el codo, abrir los dedos, coger el objeto, cerrar los dedos… Aún así, se puede dividir en problemas más pequeños, por ejemplo, ver como se mueven los motores que componen el brazo.

Problemas descomponibles.

Base de datos inicial (C, B, Z)
Operadores: R1: C → (D,L)
R2: C → (B,M)
R3: B → (M,M)
R4: Z → (B,B,M)

Resolución del problema:



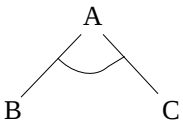
Descomponer un problema:
→ Descomponer la base de datos (estado).
→ Que los operadores se puedan aplicar a cada uno de los trozos descompuestos (en este caso, los operadores de aplican a letras, y no a cadenas, por ejemplo, asi que se puede aplicar).
→ Comprobar que el objetivo se verifica sobre las distintas de mi problema.

CONDICIONES BÁSICAS PERO NO SUFICIENTES.

Grafo Y/O

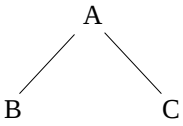
→ Descomposición de problemas: arcos Y.
→ Resolución de problemas: arcos O.
→ Concepto de solución: subgrafo.

Grafo Y: para completar el objetivo/tarea A, es necesario terminar antes los objetivos B y C.



En el cálculo proposicional, la expresión del grafo Y anterior correspondiente sería de la siguiente forma: B·C → A

Grafo O: para completar el objetivo A, es necesario terminar antes o bien el objetivo B, o bien el objetivo C.



En el cálculo proposicional, la expresión del grafo O anterior correspondiente sería de la siguiente forma: B+C → A

Grafo Y/O: combinación de grafos Y y grafos O que indican el orden de consecución de tareas a realizar para alcanzar el objetivo.

Para resolver un grafo Y/O: cada nodo se resuelve de la siguiente manera:
→ Si es un nodo Y: resolver todos sus hijos. Combinar la solución y solucionar el nodo. Devolver su solución.
→ Si es un nodo O: resolver un hijo y ver si devuelve solución. En caso contrario, resolver el siguiente hijo, etc. Cuando ya esté resuelto algún hijo, combinar la solución en el nodo y devolverla.
→ Si es un nodo terminal: resolver subproblema asociado y devolverla.

Mejora: para seleccionar el orden de resolución de nodos hijos, se puede utilizar alguna medida de estimación del coste de resolución.

Reconocimiento de frases de lengua inglesa.

Una frase está formada por un sujeto seguido de un predicado. El sujeto puede ser un sustantivo o un artículo seguido de un sustantivo. El predicado puede ser un verbo, o un verbo seguido de un complemento directo cuya estructura es idéntica a la del sujeto de la frase.

Se coge lo de la izquierda **más** lo de la derecha **obligatoriamente.** Es decir, un sujeto puede ser un nombre, o un artículo **Y** un nombre.