



Universidad de Granada

GRADO EN INGENIERÍA INFORMÁTICA

Tema 3B Inteligencia Artificial

Búsquedas con información

Fernández Fernández, Sergio

GRUPO B2

Profesor: González, Antonio

Granada, junio de 2018



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.

ESTUDIA INGLÉS
EN EL EXTRANJERO
TU FUTURO NO
TENDRÁ LÍMITES

ESTADOS UNIDOS
CANADA
REINO UNIDO
IRLANDA
NUEVA ZELANDA
AUSTRALIA

KAPLAN INTERNATIONAL
KAPLANTINTERNATIONAL.COM/ES

DESCARGA
EL CATÁLOGO
GRATUITO

KAPLANTINTERNATIONAL.COM/ES



KAPLAN INTERNATIONAL
ENGLISH

HEURÍSTICA

La heurística es un conocimiento parcial sobre un problema o dominio que permite resolver problemas eficientemente en ese problema/dominio. Las heurísticas son criterios, métodos o principios para decidir cuál de entre varias acciones promete ser la mejor para alcanzar una determinada meta. En IA, se entiende por heurística 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/experto que se tiene sobre un problema, y sirve de guía para que un algoritmo de búsqueda pueda encontrar una solución válida aceptable. Si se tiene conocimiento perfecto del problema: algoritmo exacto, si no se tiene conocimiento del problema: búsquedas sin información. Eventualmente una heurística puede devolver soluciones óptimas bajo ciertas condiciones (requiere demostración).

AJEDREZ

El jugador hace un proceso deliberativo, ¿qué puedo hacer?

8-PUZZLE

En IA, implementaremos heurísticas como funciones que devuelven un valor numérico, cuya maximización o minimización guiará al proceso de búsqueda a la solución. Tenemos que implementar una función que determine si el estado actual lleva de una manera correcta y más sencilla a la solución del problema.

Preguntas sobre heurística

1. Objetivo de una función heurística aplicada a la búsqueda en un espacio de estados

Utilizar información que disponemos acerca del problema para minimizar la búsqueda hasta el estado objetivo.

2. Heurística admisible

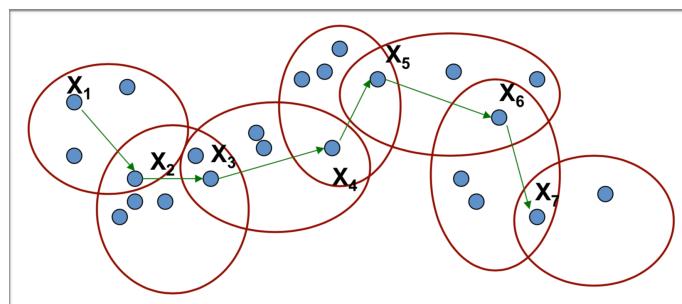
Una heurística es admisible si nunca sobreestima el costo de alcanzar el nodo objetivo. Son por naturaleza optimistas ya que piensan que el coste de resolver el problema es menor que el coste real.

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 seleccionar 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

Nos encontramos en un estado E , generamos un estado descendiente y lo comparamos con el estado actual, si es peor, generamos otro descendiente. Si el descendiente es mejor nos movemos a ese descendiente, así hasta finalizar. Si no encontramos ningún estado mejor que el primero, estaríamos en la solución, aunque no hay garantía de ello.

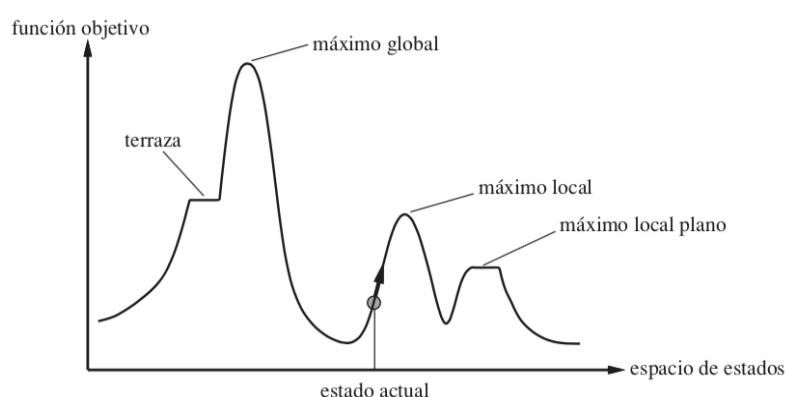


ALGORITMO DE ESCALADA POR MÁXIMA PENDIENTE

El algoritmo de escalada simple avanza cuando hay un estado mejor que el actual, en cambio este algoritmo de escalada por máxima pendiente avanza si encuentra el mejor estado de todos. Características del algoritmo:

- La solución del problema depende en gran medida de la función heurística.
- **Completitud:** no tiene por qué encontrar la solución.
- **Admisibilidad:** no siendo completo, aún menos será admisible.
- **Eficiencia:** rápido y útil si la función es monótona creciente o decreciente.
- **Solución completa:** solución correcta pero que no sabemos si es óptima.

En los métodos de escalada, los principales problemas que pueden surgir son que se encuentren un máximo local o una meseta.



ALGUNAS VARIACIONES ESTOCÁSTICAS

ALGORITMO DE ESCALADA ESTOCÁSTICO

Escoge aleatoriamente de entre los movimientos ascendentes; la probabilidad de selección puede variar con la pendiente del movimiento ascendente.

ALGORITMO DE ESCALADA DE PRIMERA OPCIÓN

Implementa una escalada estocástica generando sucesores al azar hasta que se genera uno que es mejor que el estado actual. Es una buena estrategia cuando un estado tiene muchos sucesores.

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, sino que es la mejor solución de las que se han generado.

ALGORITMO DE ENFRIAMIENTO SIMULADO

Es un método de búsqueda local que se basa en principios de la termodinámica. Al contrario que otros métodos de escalada, permite visitar soluciones peores que la actual para evitar óptimos locales.

En el campo de la termodinámica, en los años 50 se simuló el proceso de enfriamiento en sistemas de partículas hasta que se llegaba a un estado estable. El proceso simulaba la diferencia de energía del sistema.

Este proceso de enfriamiento y el algoritmo tienen mucha analogía ya que 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**, ambos tienen que minimizarse. 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 manera aleatoria, por conocimiento experto, o por medio de otras técnicas algorítmicas como greedy. La **actualización de temperatura** también es heurística. **Número de vecinos a generar** es fijo, dependiendo de la temperatura, etc.

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

Ventajas:

- Al ser un método probabilístico, tiene capacidad para salir de óptimos locales.
- Es eficiente.
- Es fácil de implementar

Inconvenientes:

- Encontrar la temperatura inicial, el método de actualización de temperatura, el número de vecinos a generar en cada estado y el número 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.

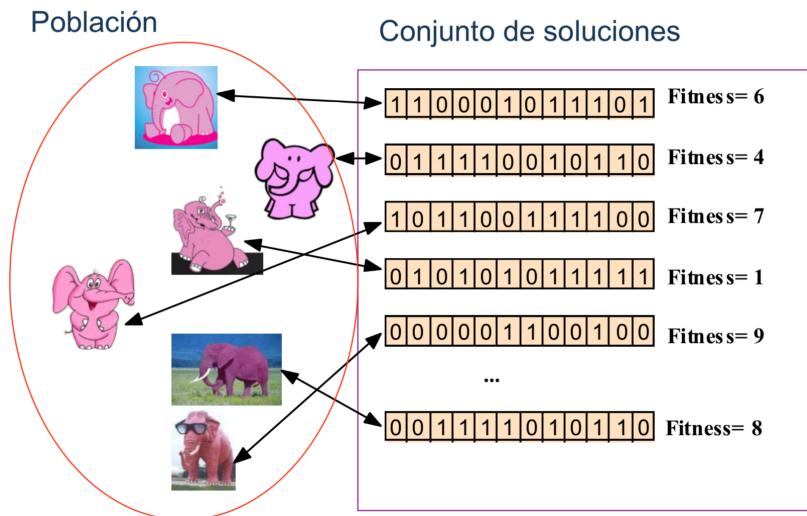
ALGORITMOS GENÉTICOS

Se va guardando una secuencia. La simulación de procesos naturales es un campo de investigación muy amplio en la IA. Los algoritmos genéticos son algoritmos de optimización basados en la teoría de la evolución natural de Darwin. En un proceso de evolución existe una **población de individuos**. Los más adecuados a su entorno **se reproducen y tienen descendencia**. Los más adecuados sobreviven para la siguiente generación. No necesitan partir de un estado o nodo inicial ya que hay una población entera. Su objetivo es encontrar una solución cuyo valor de función objetivo sea óptimo.

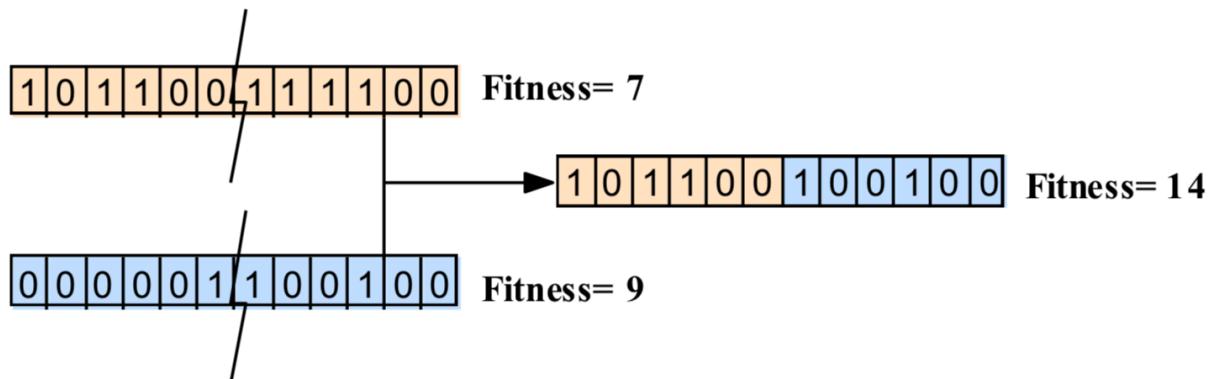
Datos:

- **Cromosoma:** vector representación de una solución al problema.
- **Gen:** característica/variable/atributo concreto del vector de representación de una solución.
- **Población:** conjunto de soluciones al problema.
- **Adecuación al entorno:** valor de función objetivo (fitness).
- **Selección natural:** operador de selección.
- **Reproducción sexual:** operador de cruce.
- **Mutación:** operador de mutación.
- **Cambio generacional:** operador de reemplazamiento.

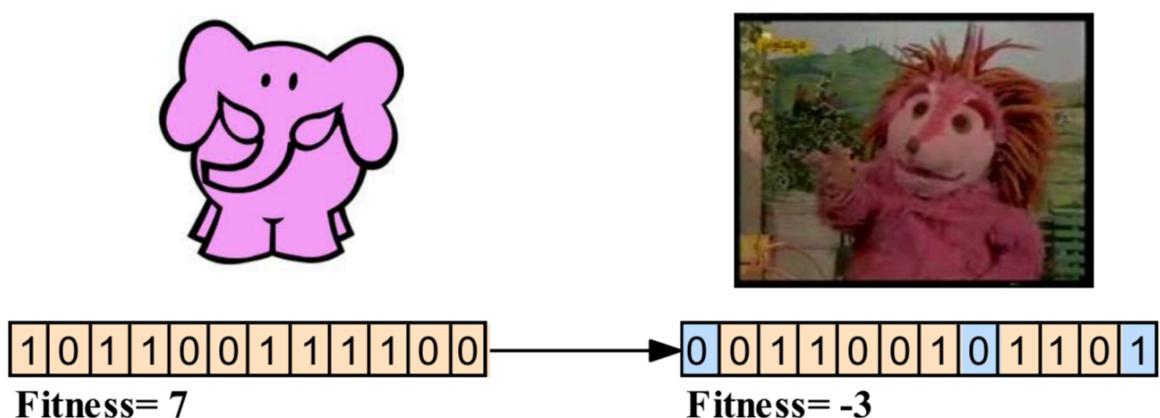
Ejemplo de población.



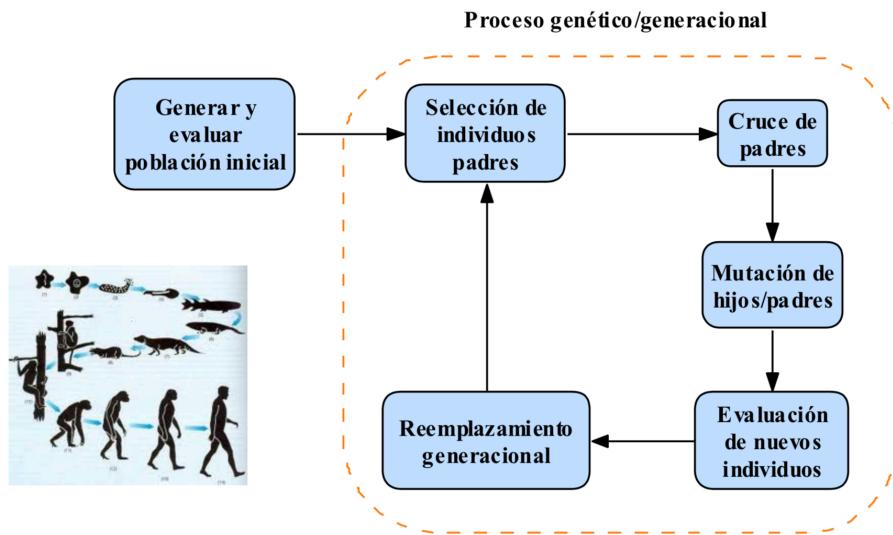
Ejemplo de cruce



Ejemplo de mutación. Uno o más genes de un individuo pueden mutar para generar una nueva solución.



Proceso de un algoritmo genético



Preguntas sobre algoritmos genéticos

1. ¿Cómo se mide la adecuación con el entorno en un algoritmo genético?

Con el valor de la función objetivo (fitness). La función de fitness se define sobre la representación genética y mide la calidad de la solución presentada. El valor de fitness determina la probabilidad de ser seleccionado.

2. ¿Qué caracteriza a los algoritmos genéticos como métodos de escalada?

(Respuesta en primer párrafo).

3. Describe como se realiza una mutación en un cromosoma en un algoritmo genético

Consiste en la modificación de un bit en el algoritmo genético. Esta modificación se efectúa con una probabilidad preestablecida, llamada probabilidad de mutación.

4. ¿Cómo se realiza el cruce entre dos nodos en un algoritmo genético?

Se selecciona un punto en el vector del primer parental. Todos los datos más allá de este punto en el vector del organismo se intercambian entre los dos organismos parentales. Los organismos resultantes son los hijos.

5. Describe como se realiza la selección de un algoritmo genético

Después de saber la amplitud de cada cromosoma se pretende elegir los cromosomas que serán cruzados en la siguiente generación. Los cromosomas con mejor amplitud tienen mayor probabilidad de ser seleccionados.

BÚSQUEDAS PRIMERO EL MEJOR

ALGORITMO A*

Función de evaluación: $f(n) = g(n) + h(n)$, donde $g(n)$ es el coste real desde el nodo raíz al nodo actual y donde $h(n)$ es el valor del nodo a evaluar desde el actual hasta el final. ABIERTOS contiene el nodo inicial, CERRADOS inicialmente 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 se termina.
- En otro caso se expande dicho nodo.
- Para cada uno de los nodos sucesores:
 - Si está en ABIERTOS insertarlo manteniendo la información del mejor parente.
 - Si está en CERRADOS insertarlo manteniendo la información del mejor parente y actualizar la información de los descendientes.
 - En otro caso, insertarlo en la lista de ABIERTOS como un nuevo nodo.

Características:

- **Completitud:** si existe solución, la encuentra.
- **Admisibilidad:** si hay una solución óptima la encuentra si:
 - El número de sucesores es finito para cada nodo.
 - $c(n_i, n_j) > \delta > 0$ en cada arco.
 - Si $g(n) = 0$, búsqueda voraz (primero el mejor).
 - Si $h(n) = 0$, búsqueda de costo uniforme.
 - La función $h(n)$ es admisible: $h(n) \leq h^*(n)$ donde $h(n)$ es la solución más corta en camino recto, $h^*(n)$ es la solución óptima y n el número de nodos.

Es un algoritmo que tiene garantías.

Preguntas sobre algoritmo A*

1. ¿Como combinación de qué dos estrategias puede verse el algoritmo A*?

Es una combinación de entre búsquedas de tipo primero en anchura (BFS) y primero en profundidad (DFS). Mientras que $h(n)$ tiende a primero en profundidad, $g(n)$ tiende a primero en anchura, de este modo se cambia de camino de búsqueda cada vez que hay nodos más prometedores.

2. A* utiliza una lista de abiertos y cerrados, describe las listas

ABIERTOS contiene los nodos que no han sido explorados todavía pero que están listos para ser explorados. CERRADOS contiene los nodos que han sido explorados y expandidos.

- 3. A* no termina mientras no se seleccione un nodo objetivo para su expansión, pero podríamos encontrar un camino al objetivo mucho antes de que se seleccione un nodo objetivo para su expansión, ¿Por qué no se termina en el momento en el que se encuentre un nodo objetivo?**

Porque podría darse la situación de encontrar un camino con menos coste si todavía no se ha expandido el nodo objetivo.

- 4. ¿Qué condiciones garantizan que el algoritmo A* obtenga la solución óptima?**

Que la heurística sea admisible, es decir, que el coste estimado debe ser menor o igual que el menor coste posible, pero en ningún caso mayor. Que el coste por arco sea positivo.

- 5. ¿Es la búsqueda de costo uniforme un caso especial de la búsqueda A*?**

Si, es un caso especial de la búsqueda A* si la heurística es una función constante. Para transformar A* en CU, hay que asumir que el costo h es constante para cualquier nodo y que el coste total dependa únicamente de g .

- 6. Interpretación de la función f en A***

$f(n) = g(n) + h(n)$ donde g es el coste desde el nodo inicial hasta n , y h es el costo heurístico desde n hasta el nodo objetivo.

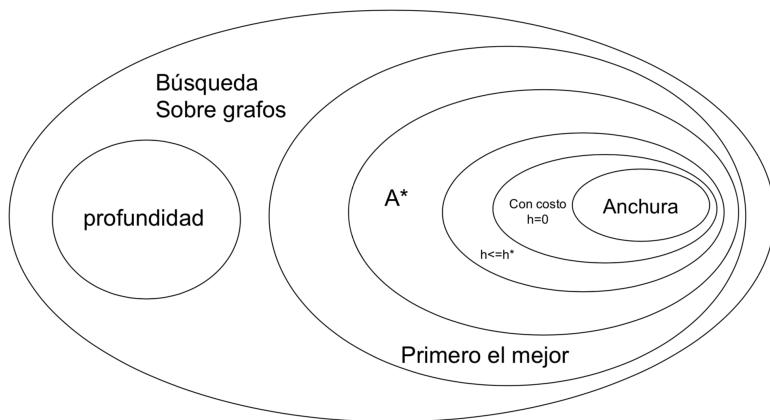
- 7. ¿Hay siempre garantía de que el algoritmo A* encuentra solución si esta existe?**

Si, en el peor de los casos y con la peor heurística, el algoritmo A* se reduce al algoritmo de búsqueda primero el mejor.

- 8. Los algoritmos de búsqueda por el mejor nodo se pueden aplicar sobre representaciones de un problema en forma de grafo o árbol. Analiza las ventajas e inconvenientes de ambas sobre el algoritmo A*.**

Sobre un grafo A* se tienen más comprobaciones puesto que hay más caminos y es más costoso para buscar una solución. Sobre un árbol A* se hacen menos comprobaciones porque solo hay un camino para llegar al nodo.

ALGORITMOS DE BÚSQUEDA

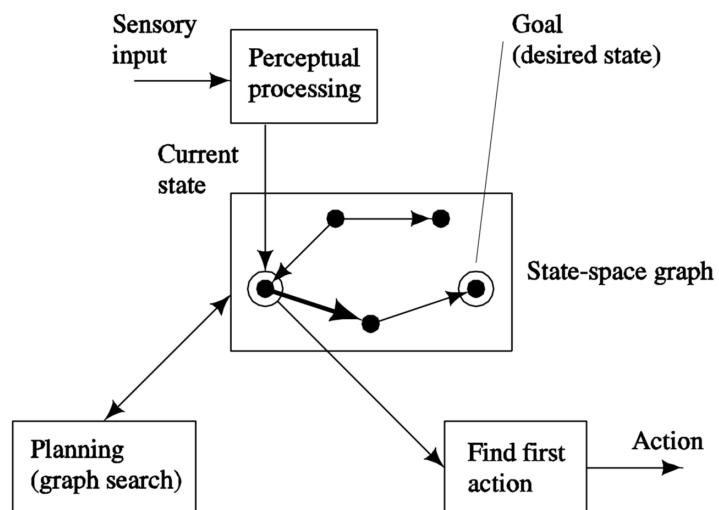


Dificultades en el proceso

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 transcurre 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 requiriese al agente actuar antes de que pudiese completar una búsqueda de un estado objetivo. Aunque el agente dispusiera de tiempo suficiente, sus recursos de memoria podrían no permitirle realizar la búsqueda de un estado objetivo.

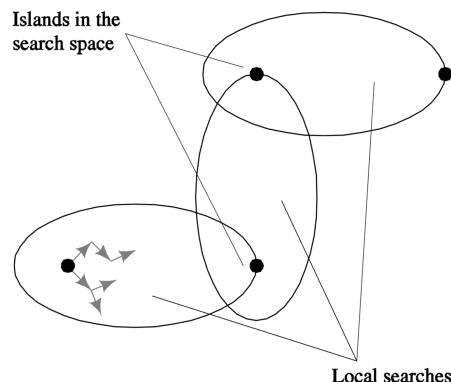
Arquitectura percepción/planificación/actuación

Poder dar respuestas 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.



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 DESCOMPOSIBLES

Base de datos inicial (C, B, Z) Operadores:

R1: C → (D,L)

R2: C → (B,M)

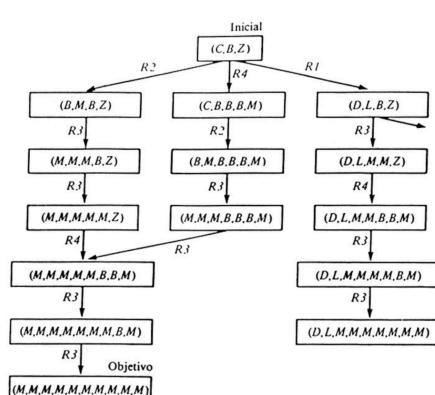
R3: B → (M,M)

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, así que se puede aplicar).
- Comprobar que el objetivo se verifica sobre las distintas de mi problema.

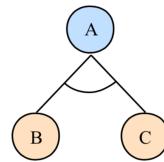


- ✓ ESCUELAS 40 ALREDEDOR DEL MUNDO
- ✓ TODOS LOS NIVELES Y OBJETIVOS
- ✓ AÑOS DE 80 EXPERIENCIA
- ✓ 97% DE RECOMENDACIÓN

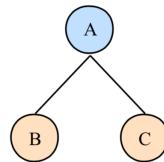
GRAFOS Y/O

Descomposición de problemas (arcos Y), resolución de problemas (arcos O), concepto de solución (subgrafo solución).

Grafo Y: para completar el objetivo/tarea **A**, es necesario terminar antes los objetivos/tareas **B** y **C**. En el cálculo proposicional, la expresión del grafo Y anterior correspondiente sería **B · C -> A**



Grafo O: para completar el objetivo/tarea **A**, es necesario terminar antes o bien el objetivo/tarea **B** o el objetivo/tarea **C**. En el cálculo proposicional, la expresión del grafo O anterior correspondiente sería **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 y devolverla
- 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 de coste de resolución.

Nueva resolución del problema

