Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda

Eduardo Morales

INAOE

Contenido

1. Introducción
2. Ejemplos e Ideas
3. Procedimientos de Búsqueda Clásicos
   - Búsqueda ciega o sin información
   - Búsqueda con Información
   - Cómo Inventar Heurísticas
   - MACRO–Operadores
4. MCTS

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Introducción

La solución de problemas está asociado a la inteligencia.
El proceso “normal” de solución de problemas involucra:

- identificación y definición del problema
- identificación del criterio de evaluación
- generación de alternativas
- búsqueda de una solución y evaluación
- selección de opción y recomendación
- implementación

En IA la solución es principalmente búsqueda y evaluación.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda

- La búsqueda es necesaria en la solución de problemas y normalmente involucra introducir *heurísticas*.
- Las heurísticas son criterios, métodos, o principios para decidir cuál de varias alternativas de acción promete ser la más efectiva para cumplir con una meta.
- Representan un compromiso entre: (i) simplicidad y (ii) poder discriminatorio entre opciones buenas y malas.
- Las heurísticas no garantizan la acción más efectiva, pero muchas veces lo hacen.
- En problemas complejos, las heurísticas juegan un papel fundamental para reducir el número de evaluaciones y para obtener soluciones dentro de restricciones de tiempo razonables.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

El problema de las 8 reinas

- Una posibilidad es lograr una solución *incremental*, acercándose a la meta poco a poco siguiendo una serie de *decisiones locales*
- Hay que asegurarse que la secuencia de transformaciones sea *sistemática*, para (i) no generar configuraciones repetidas y (ii) no excluir configuraciones deseables.
- Una forma de sistematizar la búsqueda es *construyendo* más que transformando configuraciones
- *El hecho de que podamos sistematizar la búsqueda de esta forma es que no nos podemos recuperar de violaciones a restricciones mediante operaciones futuras*.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

El problema de las 8 reinas

Posibles candidatos de heurísticas:

1. Preferir colocar reinas que dejen el mayor número de celdas sin atacar. En el ejemplo: heu1(A)=8, heu1(B)=9, heu1(C)=10.
2. Ver cuál es el menor número de celdas no atacadas en cada renglón y escoger la que su número menor sea mayor. En el ejemplo: heu2(A)=1, heu2(B)=1, heu2(C)=2

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

El problema de las 8 reinas

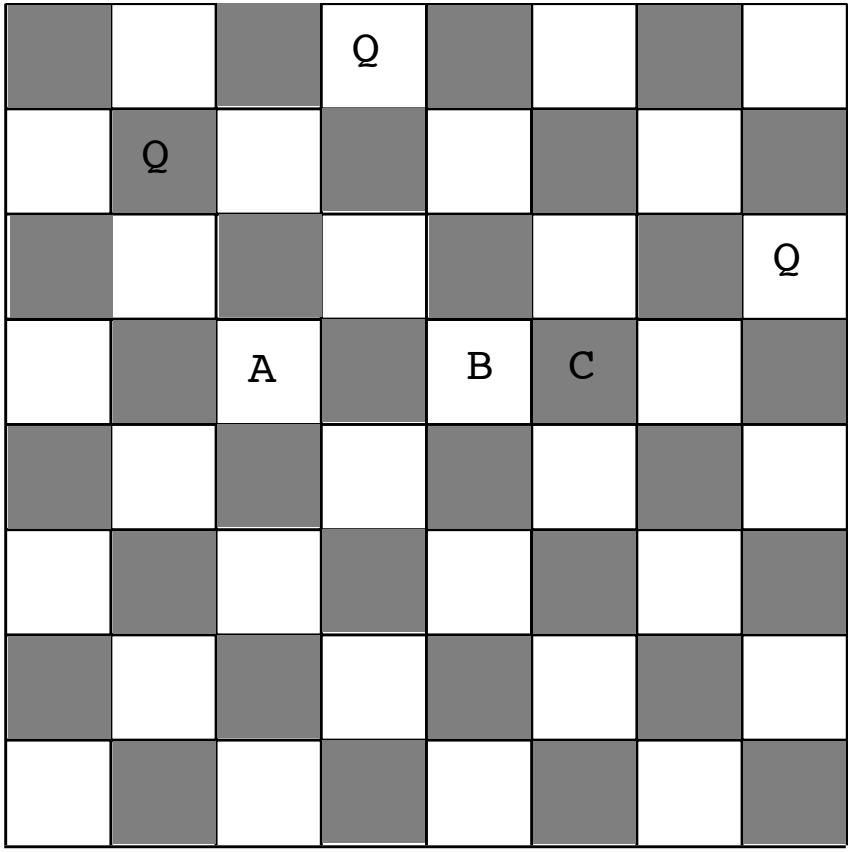


Figure: El problema de las 8 reinas

El problema del 8-puzzle

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

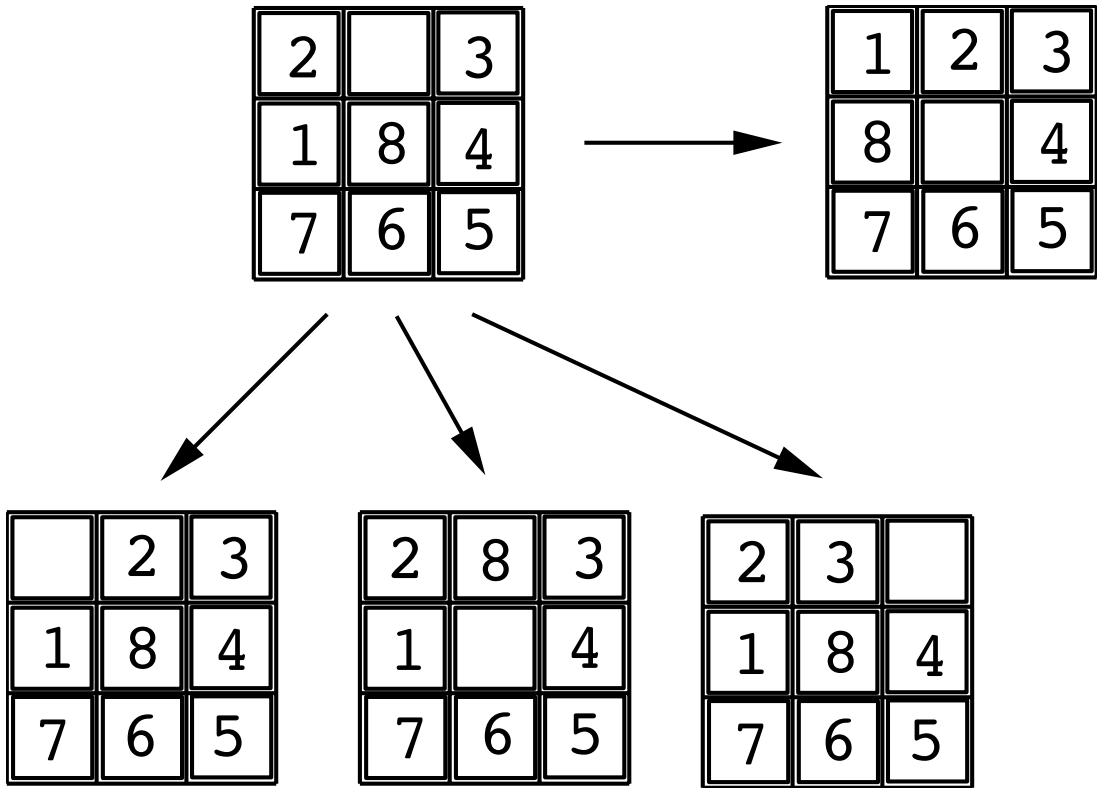


Figure: El problema del 8 puzzle

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

El problema del 8-puzzle

- Hacer *búsqueda exhaustiva* es impráctico
- Una posible regla es estimar *qué tan cerca se está de la solución*.
- Algunas heurísticas que se han usado para ésto son:
  1. Contar el número de cuadros que no corresponden a la meta (sin contar el blanco). En el ejemplo: heu1(A)=2, heu1(B)=3, heu1(C)=4.
  2. La suma de la distancia Manhattan de los cuadros que no corresponden a su lugar. En el ejemplo: heu2(A)=2, heu2(B)=4, heu2(C)=4.
  3. Distancia del cuadro blanco al primer cuadro fuera de su lugar. En el ejemplo: heu3(A)=1, heu3(B)=1, heu3(C)=3.
  4. Distancia entre la posición final del blanco y su posición actual. En el ejemplo: heu4(A)=2, heu4(B)=0, heu4(C)=2.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Encontrar la ruta más corta entre dos ciudades

- Dado un mapa con varias ciudades encontrar el camino más corto entre un par de ciudades.
- Pero si en lugar del mapa se nos dá informacíon de las distancias entre ciudades no es obvia la preferencia.
- Esto es porque en el mapa es posible estimar las distancias Euclideanas.
- Sin embargo, se pueden calcular las distancias a partir de las coordenadas de las ciudades, por lo que se puede usar esa *información extra* para determinar que acción tomar, en base a una estimación de lo que falta mas la distancia recorida.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

El problema del agente viajero

Encontrar el circuito más corto entre ciudades.

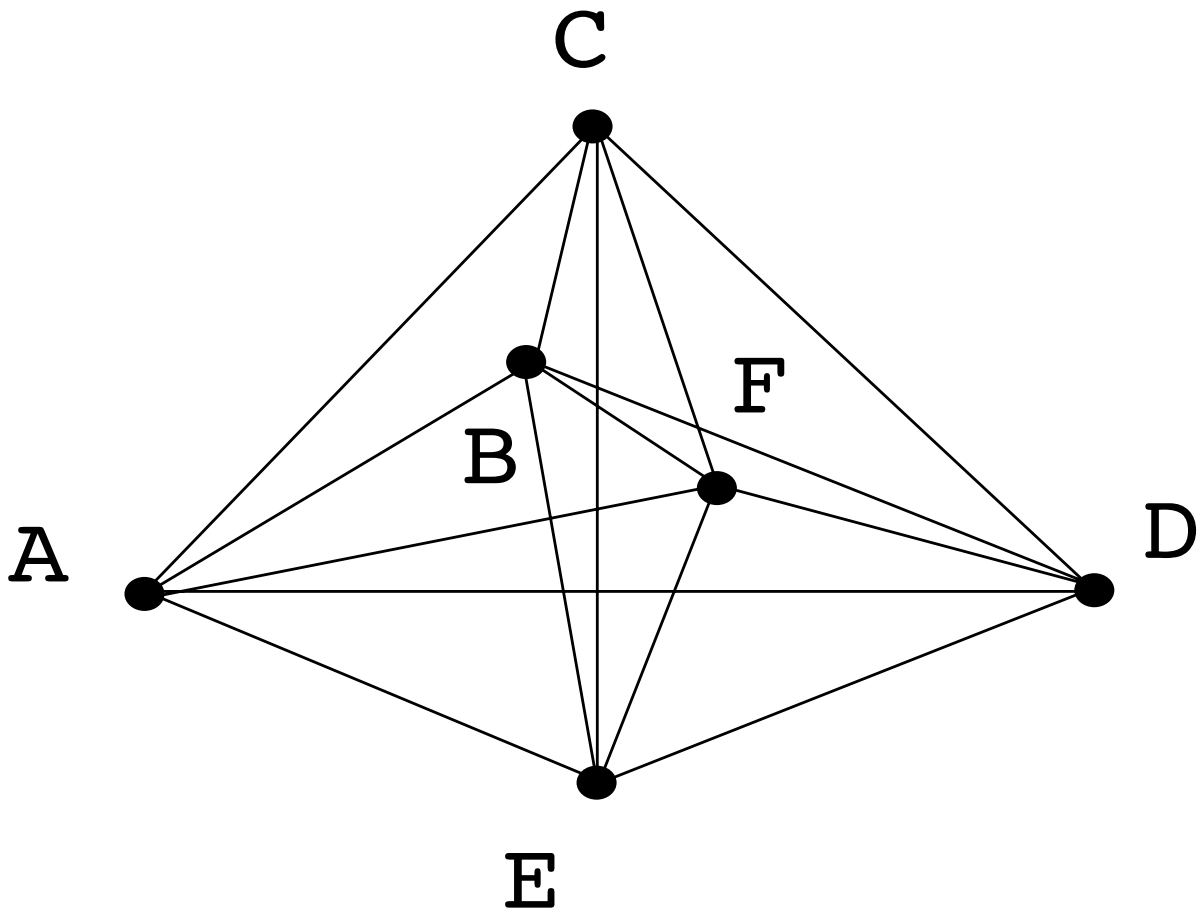


Figure: El problema del agente viajero

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

El problema del agente viajero

- Es un problema NP-duro, por lo que se requiere de *heurísticas adecuadas que acoten* el problema.
- Si tenemos un pedazo de ruta, de las diferentes alternativas que existen en esa ruta la mejor es la que nos estime completar la menor ruta.
- Esa estimación puede ser difícil de hacer, pero si es *optimísta* (subestima consistentemente el costo real) entonces el que nos de la mejor estimación es el mejor.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Detectar la moneda falsa (12 monedas, pesando 3 veces)

- Supongamos que tenemos una moneda falsa (más/menos pesada) en un grupo de 12, una balanza y tres intentos para detectarla.
- Se requiere de una *estrategia*, o sea una descripción de qué hacer primero y en base al resultado obtenido qué hacer después y así sucesivamente.
- Cada acción requiere hacer una estimación de qué hacer para las 3 posibles situaciones. Por lo que el mérito de una acción depende de la combinación de los méritos de las 3 posibles acciones (*regla de combinación*).
- Una vez que se decide alguna alternativa, se necesita especificar qué subproblema tiene que ser considerado después.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Representación

- La representación que se usa para caracterizar un problema es fundamental para su solución.
- Los requerimientos necesarios para una representación de un problema de búsqueda son:
  1. Una agenda que contiene conjuntos de soluciones potenciales
  2. Un conjunto de operaciones o reglas que nos modifican los símbolos en la agenda
  3. Una estrategia de búsqueda
- Por ejemplo, podemos jugar con un contrincante a seleccionar digitos en forma alternada, tratando de encontrar tres digitos que sumen 15. Si se usa un cuadro mágico y se juega gato el juego es trivial.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

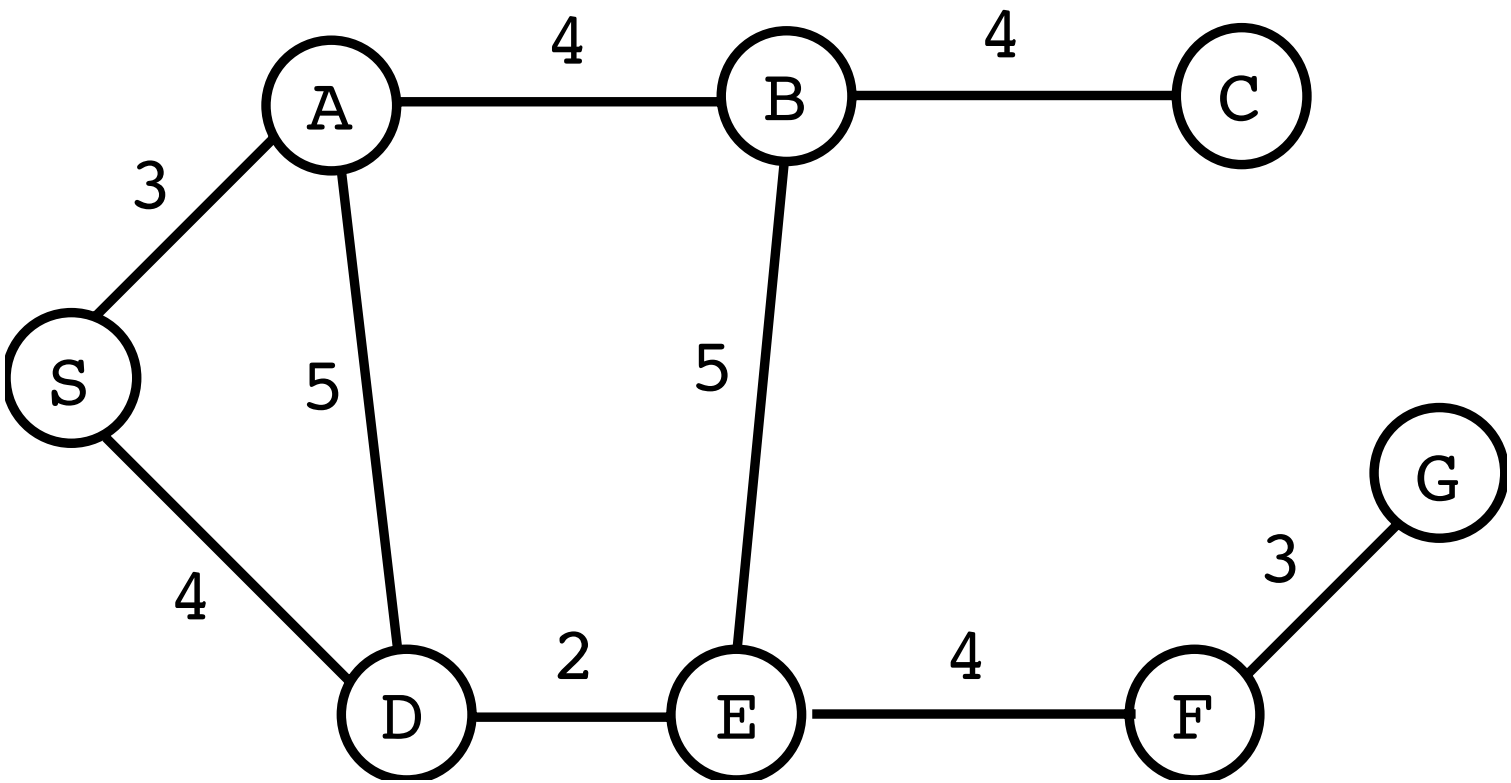
Representación

4	9	2
3	5	7
8	1	6

Figure: El cuadro mágico

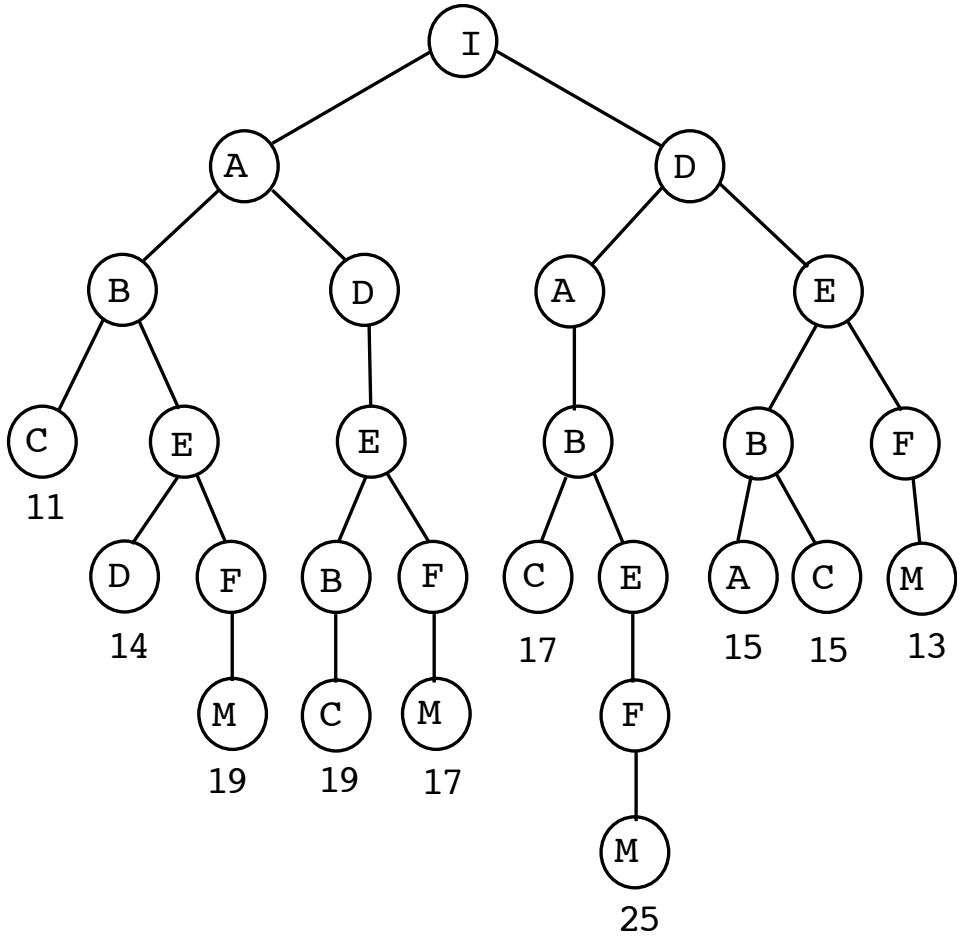
Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda Clásica



Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda Clásica



Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda Clásica

- Propiedades de algoritmos de búsqueda (heurísticas):
  1. *Completo*: un algoritmo se dice que es completo si encuentra una solución cuando ésta existe
  2. *Admisible*: Si garantiza regresar una solución óptima cuando ésta existe

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda ciega o sin información

- El orden en que la búsqueda se realiza no depende de la naturaleza de la solución buscada. La localización de la(s) meta(s) no altera el orden de expasión de los nodos.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda ciega o sin información

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta
    si el primer elemento es la meta
    entonces acaba
    si no elimina el primer elemento y
        añade sus sucesores al _____ de la agenda

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Breadth–first search (búsqueda a lo ancho)

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta
    si el primer elemento es la meta
    entonces acaba
    si no elimina el primer elemento y
        añade sus sucesores al *final* de la agenda

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Breadth–first search (búsqueda a lo ancho)

- Breadth–first es completo (encuentra una solución si existe) y óptimo (encuentra la más corta) si el costo del camino es una función que no decrece con la profundidad del nodo.
- Pero requiere de mucha memoria.
- Si se tiene un factor de arborecencia de b y la meta está a profundidad d, entonces el máximo número de nodos epandidos antes de encontrar una solución es: $1 + b + b^2 + b^3 + \ldots + b^d$

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Breadth–first search (búsqueda a lo ancho)

Profund.	Nodos	Tiempo	Memoria
0	1	1 miliseg.	100 bytes
2	111	.1 seg.	11 kilobytes
4	11,111	11 seg.	1 megabyte
6	10^6	18 min.	111 megabytes
8	10^8	31 hr.	11 gigabytes
10	10^{10}	128 días	1 terabyte
12	10^{12}	35 años	111 terabytes
14	10^{14}	3500 años	11,111 terabytes

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda de Costo Uniforme

- Una variante de breadth–first es expander todos los nodos por costos. Si el costo es igual a la profunidad se tiene el mismo algoritmo.
- La búsqueda de costo uniforme encuentra la solución más barata si el costo nunca decrece al aumentar los caminos.

$$costo(suc(n)) \geq costo(n)) \; \forall n$$

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Depth–first o Búsqueda en Profundidad (LIFO)

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta
    si el primer elemento es la meta
    entonces acaba
    si no elimina el primer elemento y
        añade sus sucesores al *frente* de la agenda

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Depth–first o Búsqueda en Profundidad (LIFO)

- Depth–first necesita almacenar un solo camino de la raíz a una hoja junto con los "hermanos" no expandidos de cada nodo en el camino.
- Por lo que con un factor de arborecencia de b y profundidad máxima de m, su necesidad de almacenamiento es a los más bm.
- Depth–first no es ni completo ni óptimo y debe de evitarse en árboles de búsqueda de profunidad muy grande o infinita.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Backtracking

- Backtracking es una versión de depth–first que aplica el criterio LIFO para generar más que para expander. Esto es, sólo se genera un sucesor a la vez.
- Su gran ventaja es su ahorro en memoria. Su gran desventaja es el no poder incluir información para evaluar cual de los sucesores es el mejor.
- Algunas modificaciones de backtracking regresan al nodo que ocaciona que se llegue a un punto sin salida (*dependency directed backtracking*).
- Backtracking también puede servir para problemas de (semi–)optimización. Si mantenemos la solución más barata hasta el momento y usamos esa información para cortar caminos (asumiendo que el costo no decrece con la profunidad de la búsqueda).

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda con Profundidad Limitada (depth–limited)

- Para evitar caer en caminos de profunidad muy grande se impone un límite de profundidad máxima.
- Si se sabe la profundidad de alguna meta, el algoritmo puede ser completo, pero sigue sin ser óptimo.
- Cuando se tienen grafos altamente conectados hay que tener cuidado con el concepto de profundidad (una más que el padre menos profundo), se tienen que analizar todos los padres, lo cual implica mas memoria.
- En la práctica se puede tomar solo la profundidad del padre que lo generó haciendo una estrategia LIFO más “pura”.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda de Profundidad Iterativa (*iterative* o *progressive deepening*)

- El problema con exigir un límite de profundidad, está precisamente en escoger un límite adecuado. Profundidad iterativa va aumentando gradualmente la profundidad hasta encontrar una meta. Es óptimo y completo como breadth–first pero requiere de poca memoria como depth–first.
- Por ejemplo, con un *branching factor* de 10 y con profundidad 5, los nodos expandidos serían $1 + 10 + 100 + \ldots + 100,000 = 111,111$, y con *iterative deepending* son: $(d+1)1 + (d)b + (d-1)db^2 + \ldots + 1 * b^d = 6 + 50 + 400 + 3,000 + 20,000 + 100,000 = 123,456 \approx 11\%$ más de nodos.
- Entre más grande sea el factor de arborecencia, menor el trabajo extra.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda Bidireccional

- Aquí la idea es explorar al mismo tiempo del estado inicial hacia la meta que de la meta hacia el estado inicial hasta que se encuentren en un estado.
- Cuando el factor de arborecencia es igual en ambos sentidos, se pueden hacer grandes ahorros.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Puntos a Considerar

- Cuando los operadores son reversibles, los conjuntos de predecesores y sucesores son iguales, pero en algunos casos calcular los predecesores puede ser muy difícil.
- Si hay muchas posibles metas explícitas, en principio se podría hacer la búsqueda hacia atrás a partir de cada una de ellas. Sin embargo, a veces las metas son sólo implícitas (e.g., jaque mate).
- Se necesita un método eficiente para revisar si un nuevo nodo aparece en la otra mitad de la búsqueda.
- Se tiene que pensar que tipo de búsqueda hacer en cada mitad (no necesariamente la misma es la mejor).

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Comparación de Algoritmos

Comparación de algoritmos. b = factor de arborescencia, d = profundidad de la solución, m = profundidad máxima, l = profundidad límite.

Criterio	Breadth first	Costo unif.	Depth first	Depth limited	Itera. deep.	Bidir.
Tiempo	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Espacio	b^d	b^d	$b \times m$	$b \times l$	$b \times d$	$b^{d/2}$
Optimo	si	si	no	no	si	si
Completo	si	si	no	si (si $l \geq d$)	si	si

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Algoritmo Genérico con Información

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta
    si el primer elemento es la meta
    entonces acaba
    si no elimina el primer elemento,
        añade sus sucesores a la agenda,
        ordena todos los elementos de la agenda

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Hill–Climbing

Hill–climbing es una estrategia basada en optimización local.

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta
si el primer elemento es la meta
entonces acaba
si no elimina el primer elemento,
añade sus sucesores a la agenda,
ordena todos los elementos de la agenda
selecciona el mejor y *elimina el resto*

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Hill-Climbing

- Problemas obvios: máximos/mínimos locales, valles y riscos
- Para salir de minimos/máximos locales o tratar de evitarlos con hill–climbing a veces se empieza la búsqueda en varios puntos aleatorios (*random–restart hill–climbing*), salvando el mejor
- Dado su bajo costo computacional es la estrategia de búsqueda más utilizada en optimización y aprendizaje.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Best–First

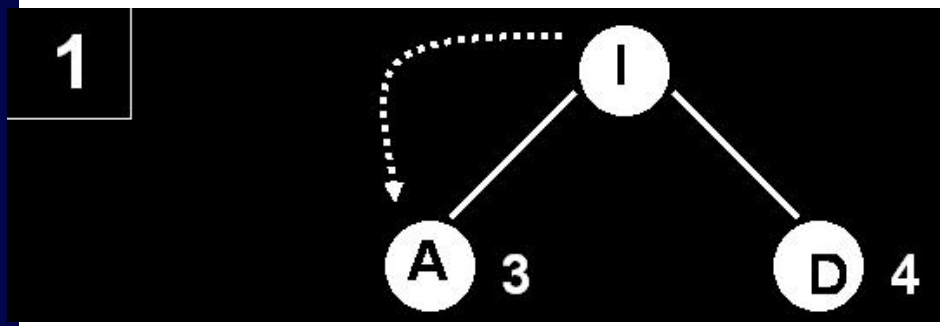
Si el nodo que mejor evaluación recibe es el que se expande primero, entonces estamos haciendo “best–first”.

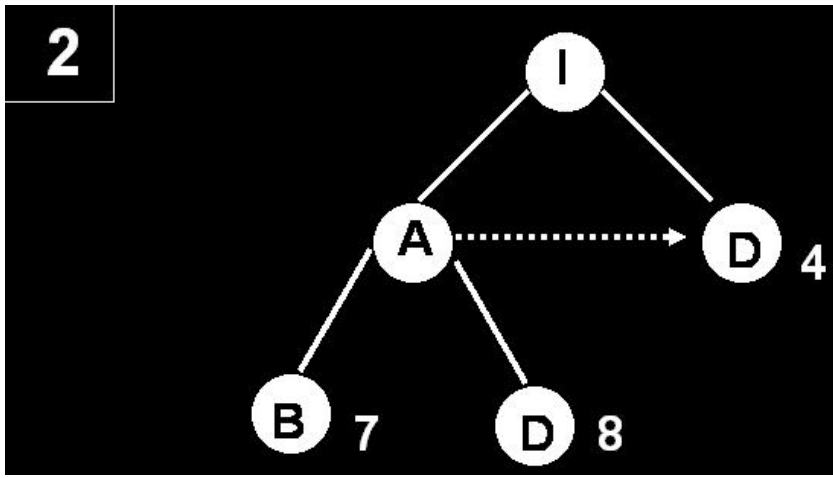
Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta
si el primer elemento es la meta
entonces acaba
si no elimina el primer elemento,
añade sus sucesores a la agenda,
ordena todos los elementos de la agenda

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Best-First

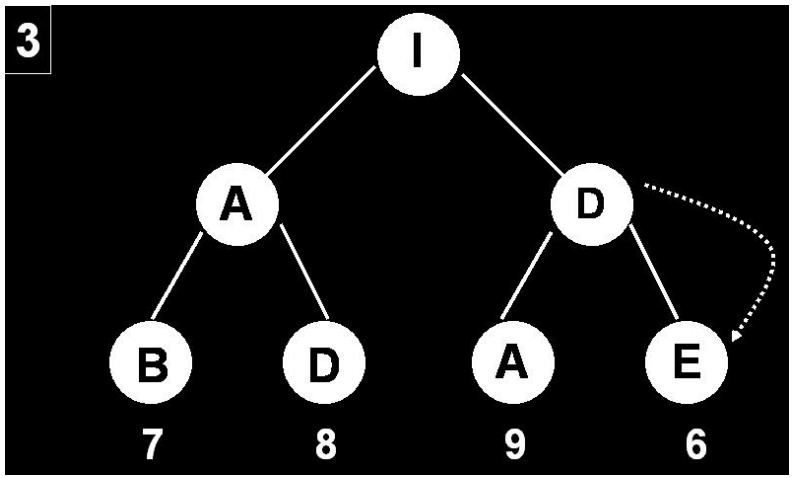
Más que estar expandiendo el “mejor”, se expande el que parece ser el mejor de acuerdo con nuestra función de evaluación.

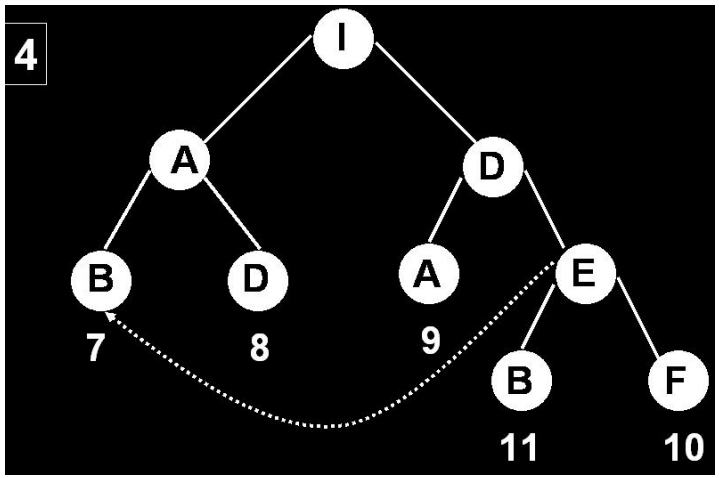




Ejemplo

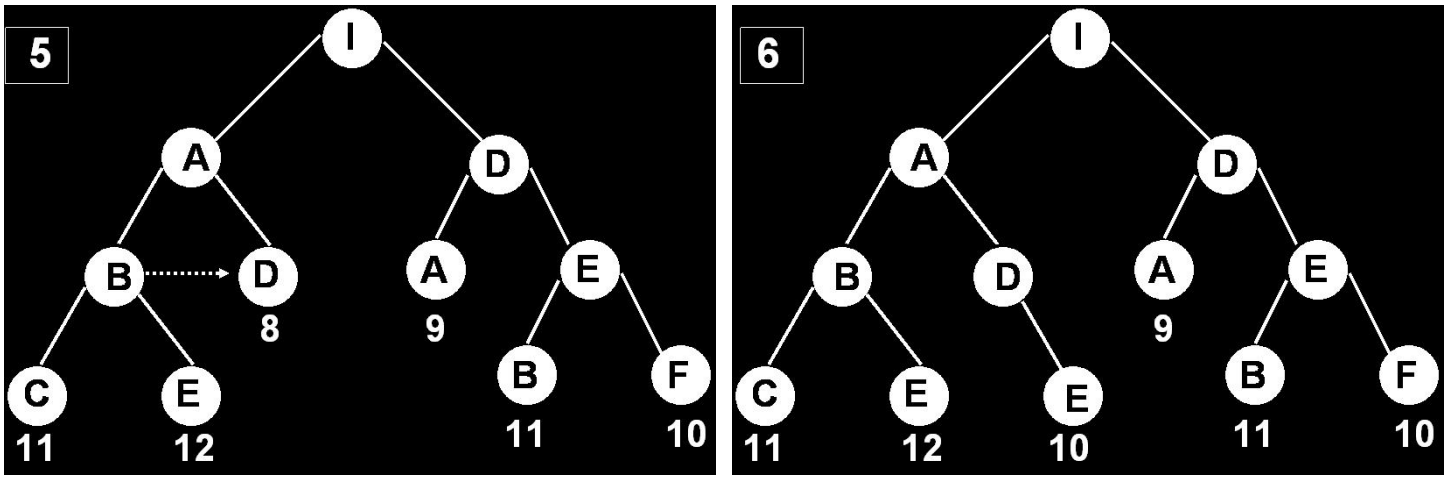
Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS





Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Ejemplo



Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Ejemplo

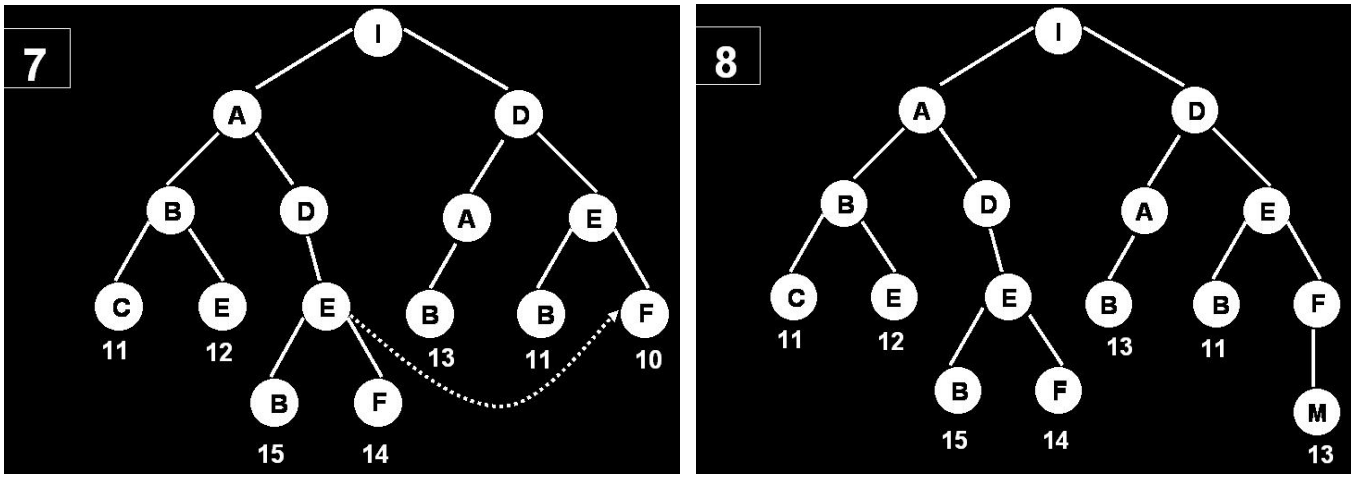


Figure: Búsqueda Best first

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Usando Costo Estimado

- El usar el costo acumulado (búsqueda de costo uniforme) no necesariamente guía la búsqueda hacia la meta. Para ésto, se utiliza una estimación del costo del camino del estado hacia una meta ($h(n)$).
- Esta estrategia (minimizar el costo estimado para alcanzar una meta) a veces se llama una estrategia "greedy".
- La función de evaluación (h) puede ser lo que sea mientras $h(n) = 0$ si n es una meta.
- Debido a que guardan todos los nodos en memoria, su complejidad en espacio es igual a la del tiempo.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Híbridos

- Tres de las estrategias principales vistas: hill–climning, backtracking y best–first, se pueden ver como 3 puntos en un espectro continuo de estrategias, si las caracterizamos por:
  - Recuperación de caminos sin salida
  - Alcance de evaluación (número de alternativas consideradas)

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Híbridos

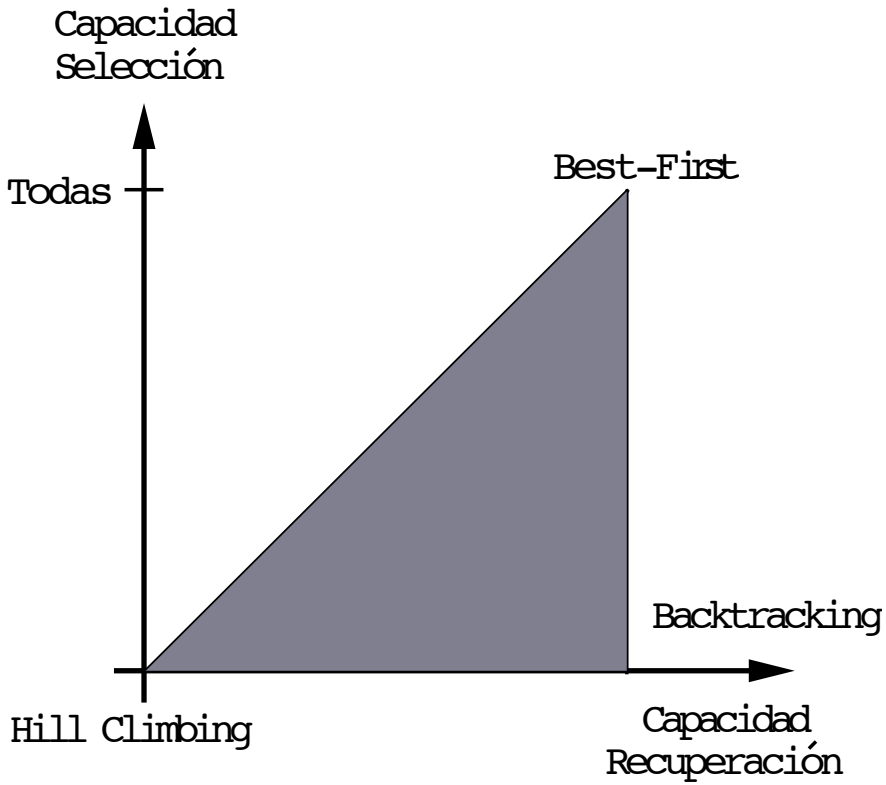


Figure: Algoritmos Híbridos

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Híbridos

- Se puede hacer una combinación BF–BT. Se aplica BF al principio hasta agotar la memoria, seguido de BT. Si BT no encuentra una solución, se considera otro nodo.
- Otra posibilidad es usar BT hasta cierta profundidad y luego emplear BF. Si no se llega a una solución hacemos backtracking y buscamos en otra zona.
- Esta segunda opción es más difícil de controlar que la primera, pero aplicar BF al final tiene la ventaja que BF se comporta mejor entre más informado esté (lo cual es más probable que ocurra en la parte inferior del árbol)

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Híbridos

- Se puede tener una combinación de un BF local con un BT global. Se empieza con un BF con memoria limitada. Cuando se acaba la memoria, su lista de nodos en OPEN se toma como los sucesores directos del nodo de donde se partió (el nodo raíz la primera vez), los cuales se someten a un BT. Sin embargo, cada uno de ellos se expande usando un BF con memoria limitada. Si no se llega a la solución se hace backtracking a otro nodo.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

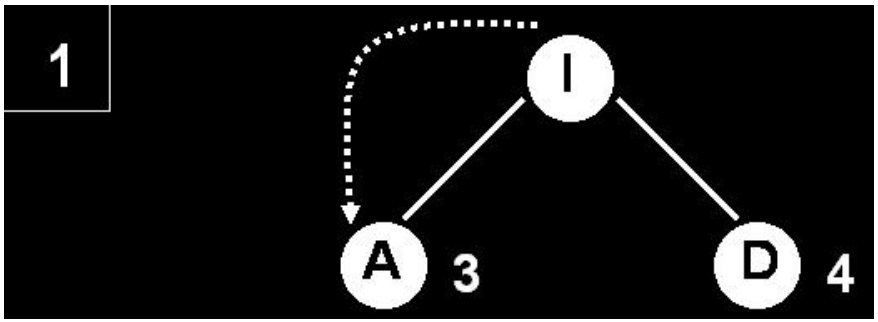
Beam–Search

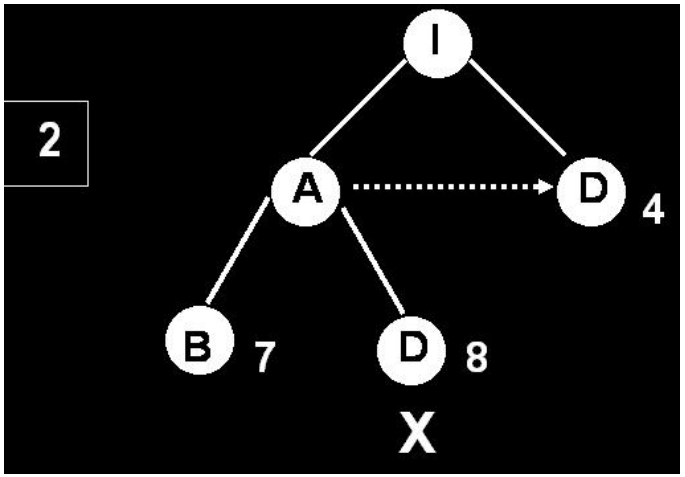
Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta
    si el primer elemento es la meta
    entonces acaba
    si no elimina el primer elemento,
        añade sus sucesores a la agenda,
        ordena todos los elementos de la agenda y
        selecciona los k mejores (elimina los demás)

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Beam–Search

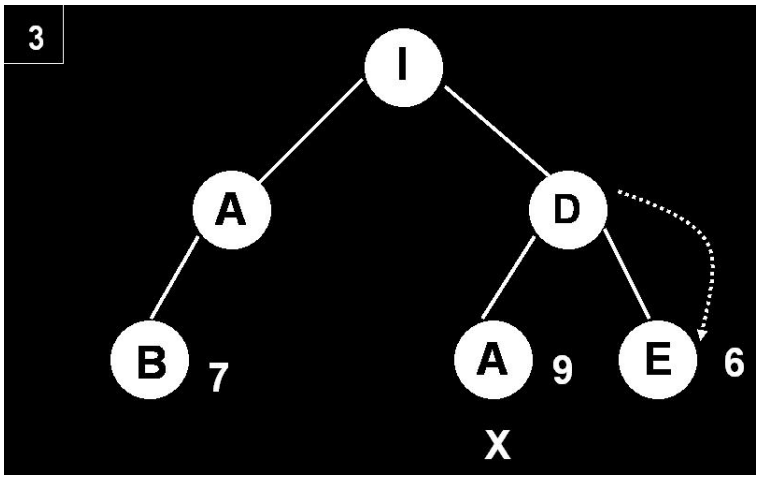
Si $k = 1$ es como hill-climbing, si k es tan grande para considerar todos los nodos de la agenda es como best-first.

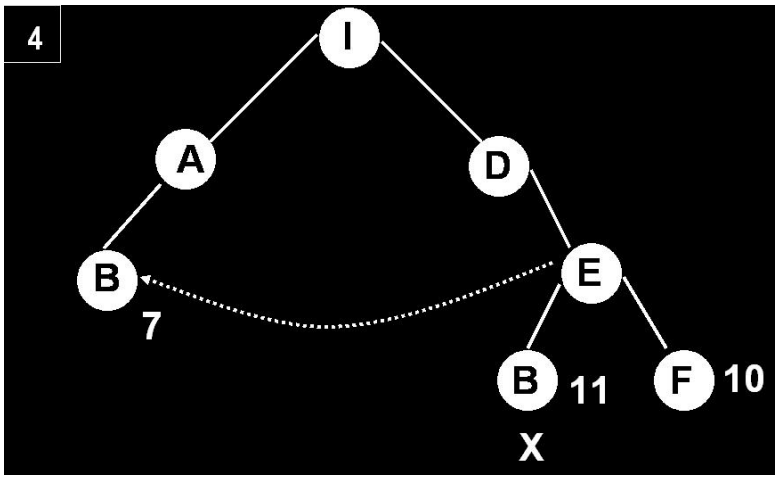




Beam-Search

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS





Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Ejemplo

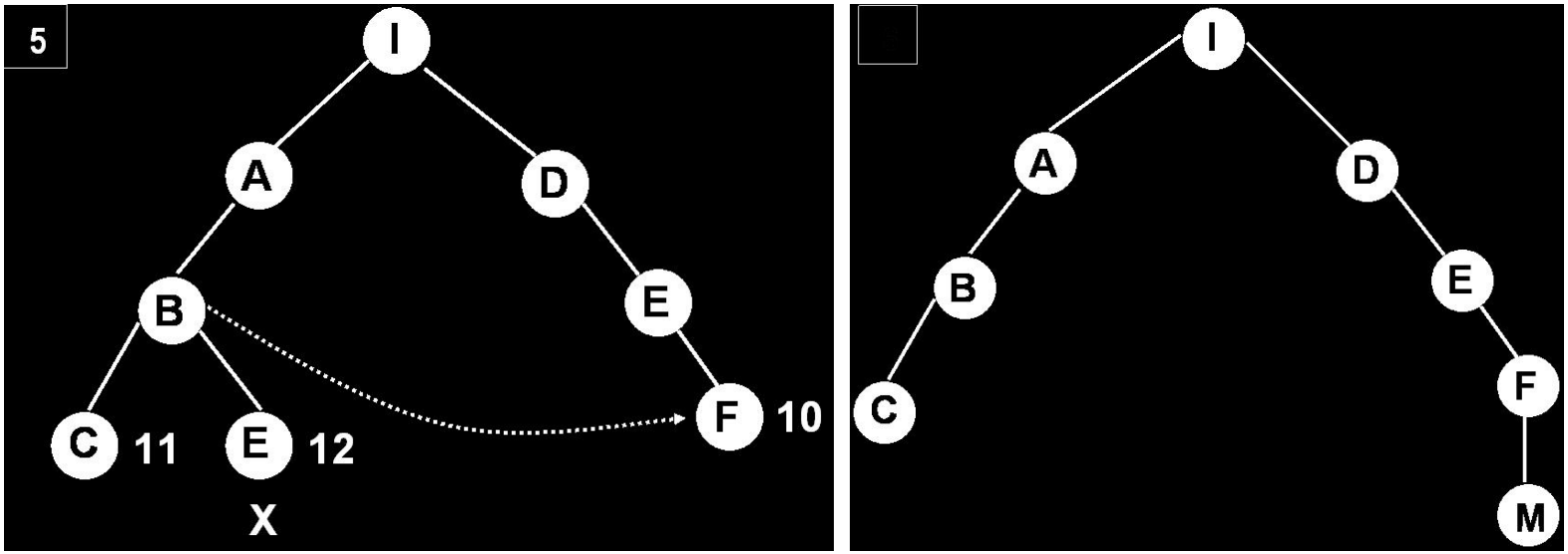


Figure: Búsqueda Beam search con $k = 2$.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

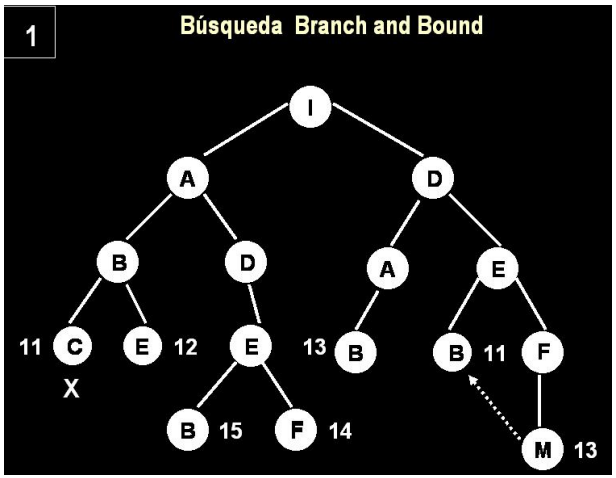
Branch and Bound

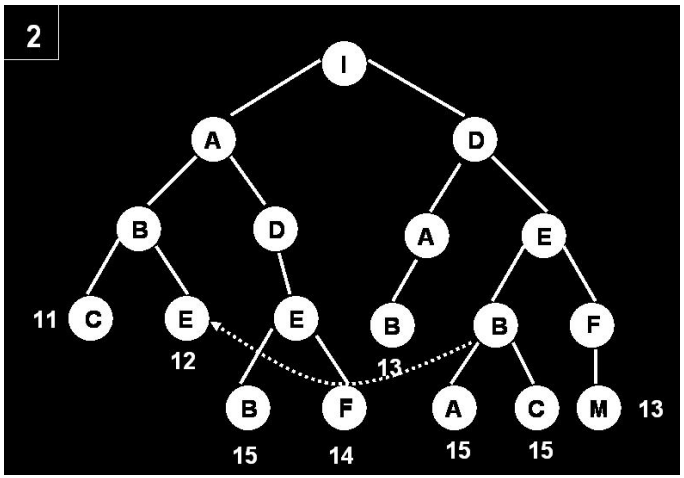
Trabaja como best-first pero en cuanto se encuentra una solución sigue expandiendo los nodos de costos menores al encontrado

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta y los demas nodos sean de costos mayores o iguales a la meta
si el primer elemento es la meta y los demas nodos son de menor o igual costo a la meta
entonces acaba
si no elimina el primer elemento y añade sus sucesores a la agenda
ordena todos los elementos de la agenda

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Ejemplo





Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Ejemplo

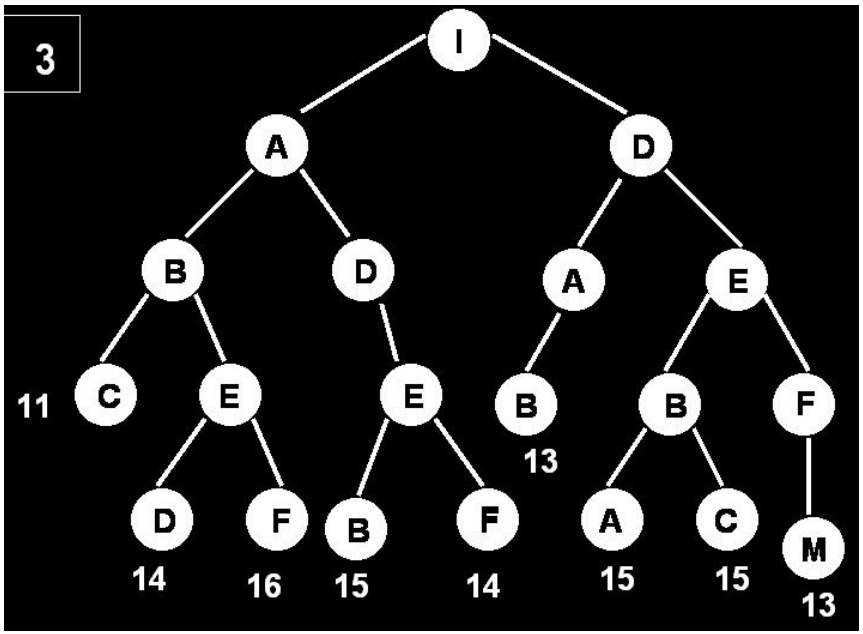


Figure: Búsqueda Branch and Bound

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

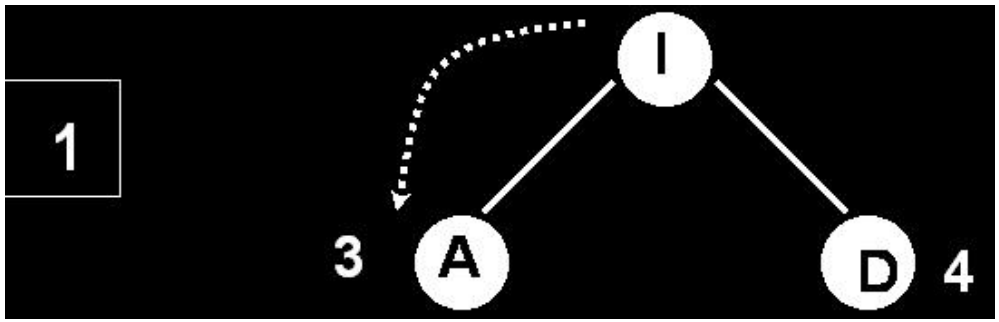
Dynamic Programming

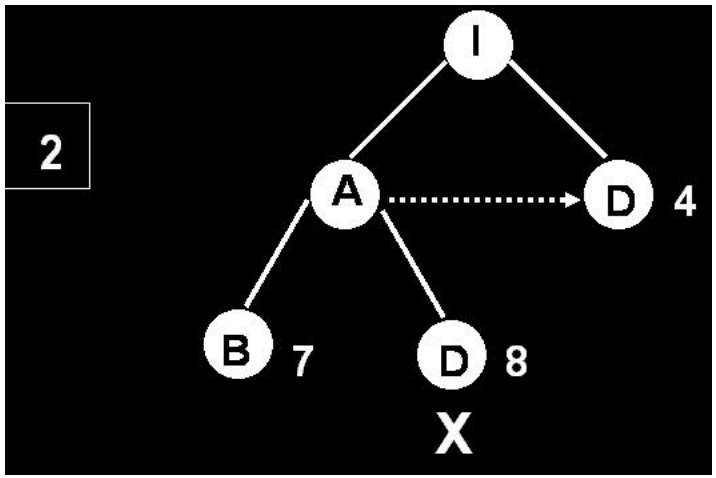
- Idea: no explorar caminos a los que yá llegamos por caminos más cortos/baratos.
- El algoritmo es igual sólo hay que añadir la condición:

  elimina todos los caminos que lleguen al mismo nodo excepto el de menor costo

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

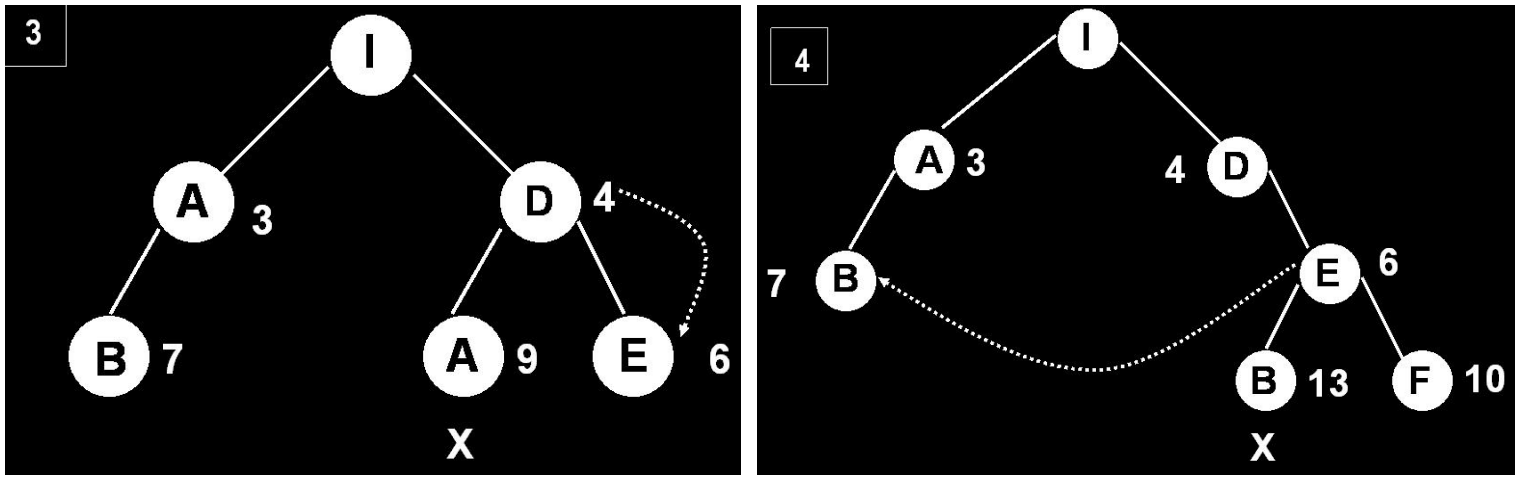
Ejemplo





Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Ejemplo



Ejemplo

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

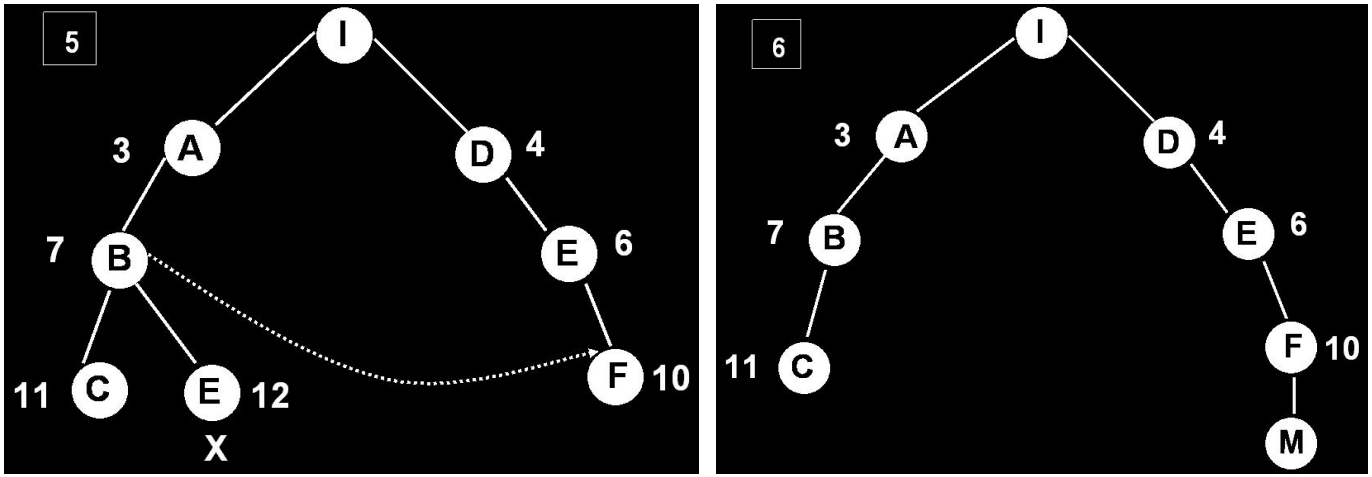


Figure: Búsqueda con programación dinámica

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

A*

- Búsqueda "greedy" minimiza $h(n)$ reduciendo la búsqueda, pero no es ni óptima ni completa. Búsqueda de costo uniforme minimiza el costo acumulado $g(n)$ y es completa y óptima, pero puede ser muy ineficiente.
- Se pueden combinar las dos sumandolas, usando para cada nodo la siguiente heurística: $f(n) = g(n) + h(n)$.
- La estrategia se puede probar que es completa y óptima si $h(n)$ nunca sobre–estima el costo. Tal función se le conoce como una heurística *admisible*.
- Breadth–first se puede ver como un caso especial de A* si $h = 0$ y $c(n, n') = 1$ para todos sus sucesores (el costo de ir de un nodo padre a un nodo hijo).
- Si h es admisible, $f(n)$ nunca hace sobrestimaciones.

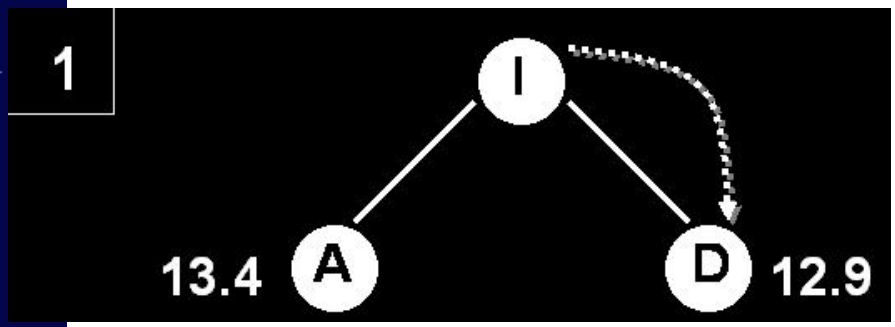
Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

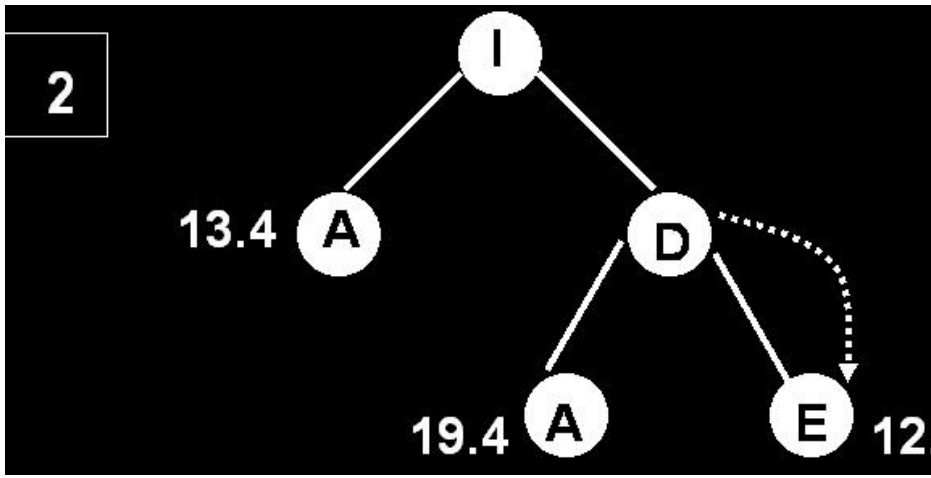
Algoritmo A*

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta y los demas nodos sean de costos mayores o iguales a la meta
si el primer elemento es la meta y los demas nodos son de menor o igual costo a la meta
entonces acaba
si no elimina el primer elemento y añade sus sucesores a la agenda
ordena todos los elementos de la agenda de acuerdo al costo acumulado más las subestimaciones de los que falta
elimina todos los caminos que lleguen al mismo nodo, excepto el de menor costo

Ejemplo

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS





Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Ejemplo

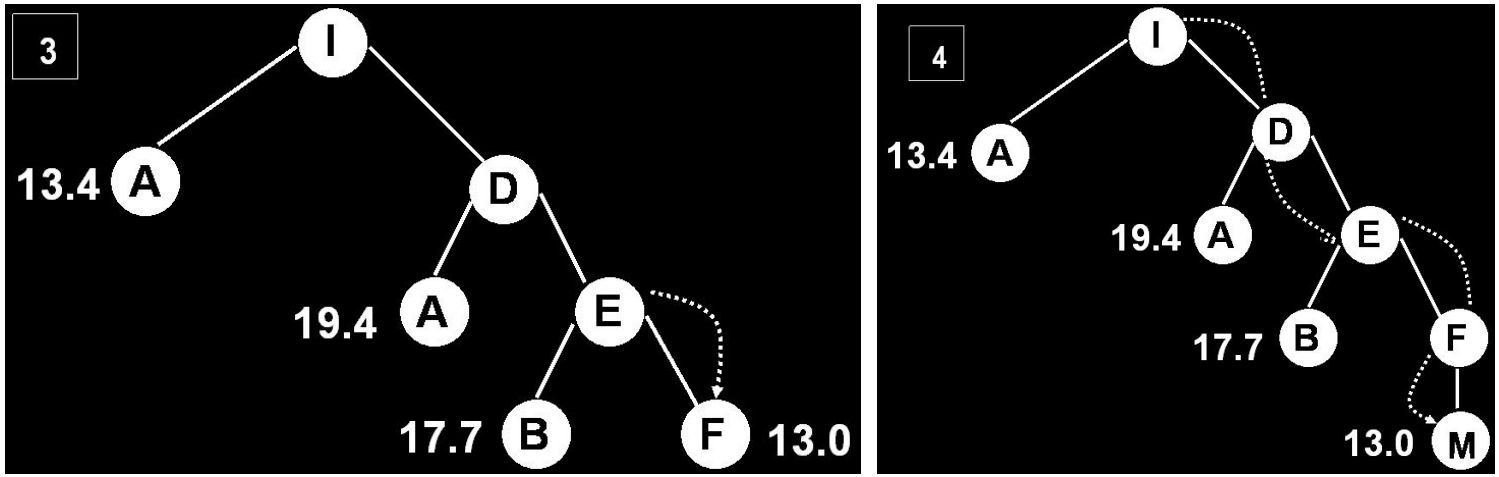


Figure: Búsqueda A*

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Comparación

Resultados promedios de 100 instancias del *8-puzzle*.

	Costo de búsqueda			Arborecencia efetiva		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Extensiones

- $f = g + h$ viene de dos principios: (i) lo más corto es lo más rápido (g) y (ii) para asegurarnos que es la mejor opción hay que agregar subestimaciones (h).
- El efecto de g es para añadir la componente de breadth–first a la búsqueda, mientras h es ideal cuando se tienen muchas soluciones y se puede confiar en la estimación.
- Se han propuestos funciones pesadas para compensar cada parte:

$$f(n) = (1 - w)g(n) + wh(n)$$

$w = o \Rightarrow$ búsqueda de costo uniforme
$w = 1/2 \Rightarrow$ A*
$w = 1 \Rightarrow$ BF

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Extensiones

- Una estrategia ϵ-admisible se obtiene con un peso dinámico: En lugar de mantener a w constante durante la búsqueda, mejor usar un peso dinámico que se ajusta con la profundidad. Una posibilidad es:

$$f(n) = g(n) + h(n) + \epsilon[1 - \frac{d(n)}{N}]h(n)$$

  donde $d(n)$ es la profundidad del nodo n y N la profundidad anticipada del nodo meta. A profunidades bajas, h tiene un soporte de $1 + \epsilon$ lo que favorece escursiones tipo depth-first, mientras que a grandes profundidades asume un peso igual.
- Se pueden usar otras posibles medidas no aditivas para h y g: e.g., multiplicativa.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

IDA*

- Generalmente lo primero que se agota en A* es la memoria. Dos extensiones a A* tratan de reducir los requerimientos de memoria.
- IDA*, es una extensión natural de *iteractive deepening*. Se usa un límite de costo (más que de profundidad) iterativo.
- IDA* es completo y óptimo como A*, pero por ser depth–first, sólo requiere memoria proporcional al camino más largo que explora (aprox. *bd*, porqué depende del número de nodos por costo).

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

IDA*

- Sin embargo, en problemas en donde la heurística cambia para cada estado (e.g., el TSP), lo que quiere decir es que cada contorno incluye sólo un estado. Si A* expande N nodos, IDA* tendría que hacer N iteraciones $(1 + 2 + \ldots + N)$, osea $O(N^2)$
- Por lo que si N es muy grande para ser guardada en memoria, seguramente N^2 es demasiado tiempo que se necesita esperar.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

A^*_ϵ

- Cuando se tienen muchas posibilidades más o menos todas del mismo costo, lo que se puede hacer es agruparlas todas en un mismo rango y sólo expander la mejor de acuerdo a cierta heurística de simplicidad de solución, permitiendo errores del orden del rango establecido.
- Es como A*, sólo que ahora se tienen dos listas: OPEN (como antes) y FOCAL $\subset$ OPEN, en donde se tienen todos los nodos que no se desvían de la mejor opción en más de ϵ.
- Se comporta como A* pero selecciona un nodo de FOCAL usando una heurística nueva

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MA*

- Espacios de estados con una gran variedad de costos son difíciles porque el algoritmo extiende su frontera y cambia frecuentemente de oponión acerca de qué camino es el más promisorio.
- Para IDA* es especialmente difícil porque no guarda información acerca de los caminos.
- MA* al llenar su memoria elimina el nodo de mayor costo de su agenda y propaga su costo a su nodo padre.
- A pesar de eliminar la información que dice como ir a un cierto nodo, los valores guardados le sirven al algortimo para saber que tanto vale la pena explorar el nodo padre.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Mejoras Dinámicas

- Las evaluaciones en las heurísticas son básicamente estáticas e involucran cierto error (e.g., subestimaciones).
- Una forma de mejorar las funciones de evaluación es actualizarlas dinámicamente conforme se obtiene más informacón.
- Algunas variantes incluyen (aunque existen otras):
  1. *Add method*
  2. *Max method*
  3. *Back-up method*
  4. *Front-to-front method*

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Add Method

1. Se calcula alrededor de la meta rutas a varios puntos (B_is),
2. Se estima el error entre el valor real de todos estos puntos ($g*(B_i)$) y el estimador inicial a estos puntos ($Diff(B_i) = g*(B_i) - h(B_i)$),
3. se toma el error mínimo ($minDiff = min_i(Diff(B_i)$ y se usa para mejorar el estimador de los nodos (As) haciendo búsqueda hacia adelante ($newh(A) = h(A) + minDiff$).

Add Method

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

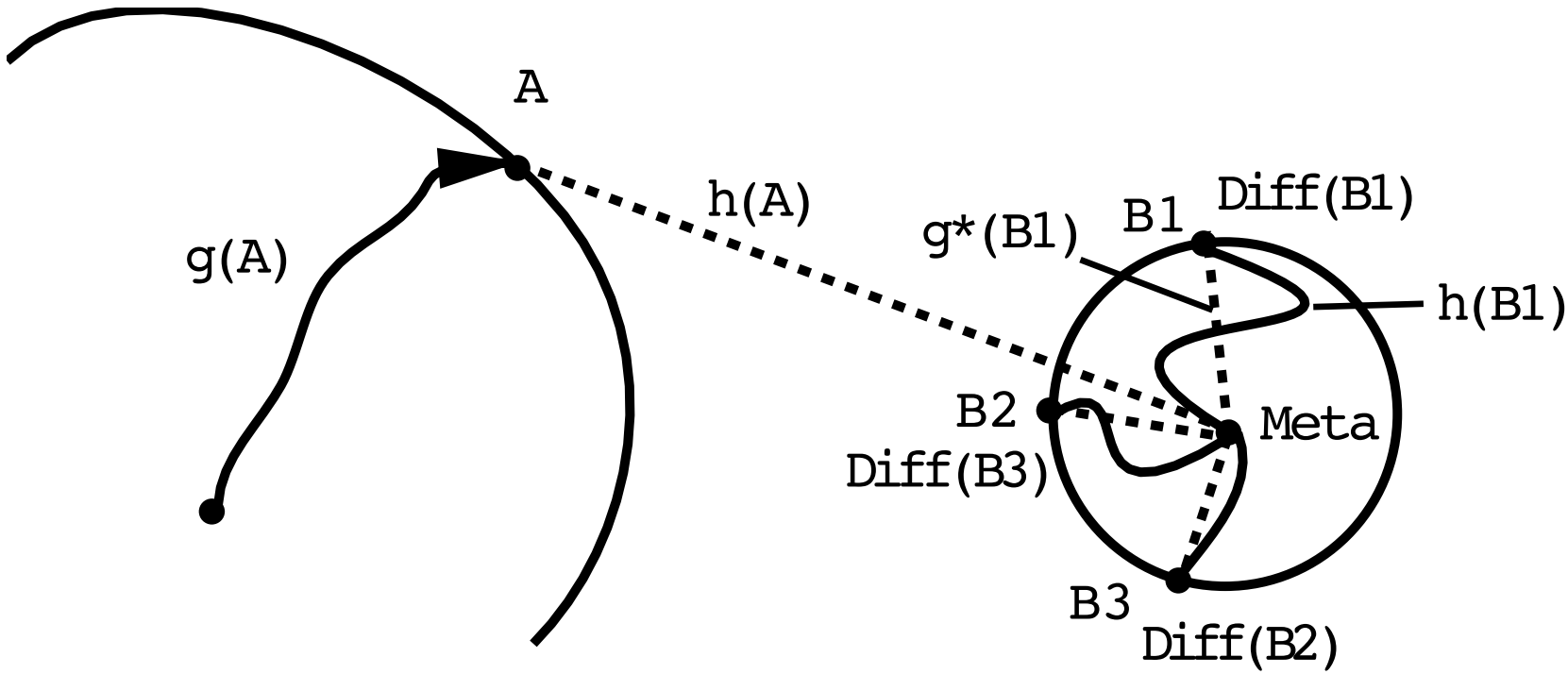


Figure: Esquema ilustrando el método *Add*

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Max Method

1. Se calcula alrededor de la meta rutas a varios puntos ($B_i s$),
2. Se estima la distancia entre el estado inicial y los puntos cercanos a la meta y se estima el mínimo de la distancia total ($fmin2 = min_i(g * (B_i) + h(B_i))$),
3. se usa para mejorar el estimador de los nodos (As) haciendo búsqueda hacia adelante, tomando en cuenta el estimador inicial hacia A, $h_1(A)$ ($h'(A) = fmin2 - h_1(A)$),
4. como no siempre es mejor estimador que el que se tenía, se hace la combinación max de ambos ($newh(A) = max(h(A), fmin2 - h_1(A))$.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Max Method

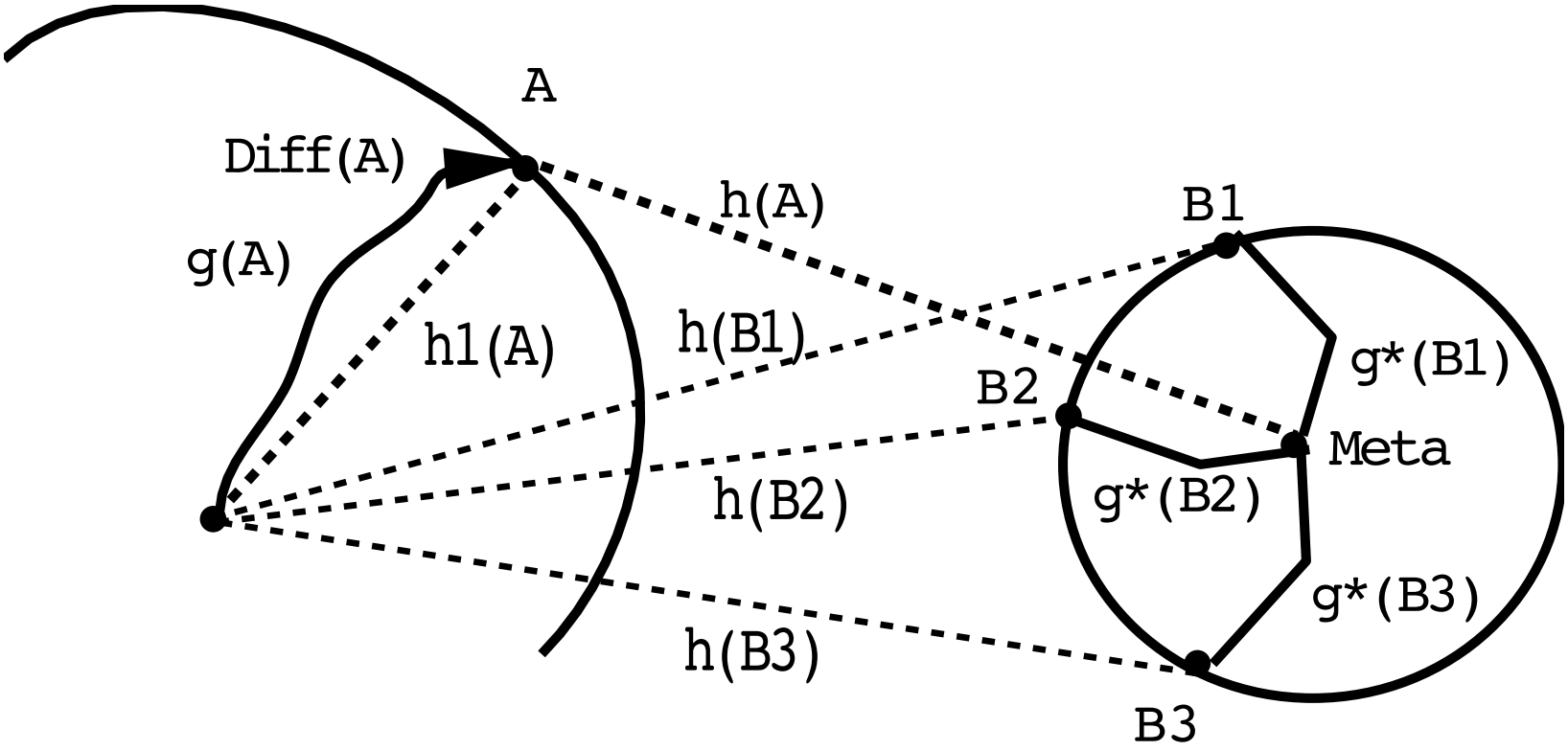


Figure: Esquema ilustrando el método *Max*

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Back-up method

1. Se toman los sucesores de un nodo A (Bs),
2. Se estima el costo de A a cada uno de los Bs ($k_i(A, B_i)$) y la estimación de cada B a la meta ($h(B_i)$),
3. Se usa como estimador de A a la meta, el mínimo de estos ($newh(A) = min_i(k_i(A, B_i) + h(B_i))$). Si el evaluador no es consistente, entonces se toma: $newh(A) = max(h(A), min_i(k_i(A, B_i) + h(B_i)))$.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Back-up Method

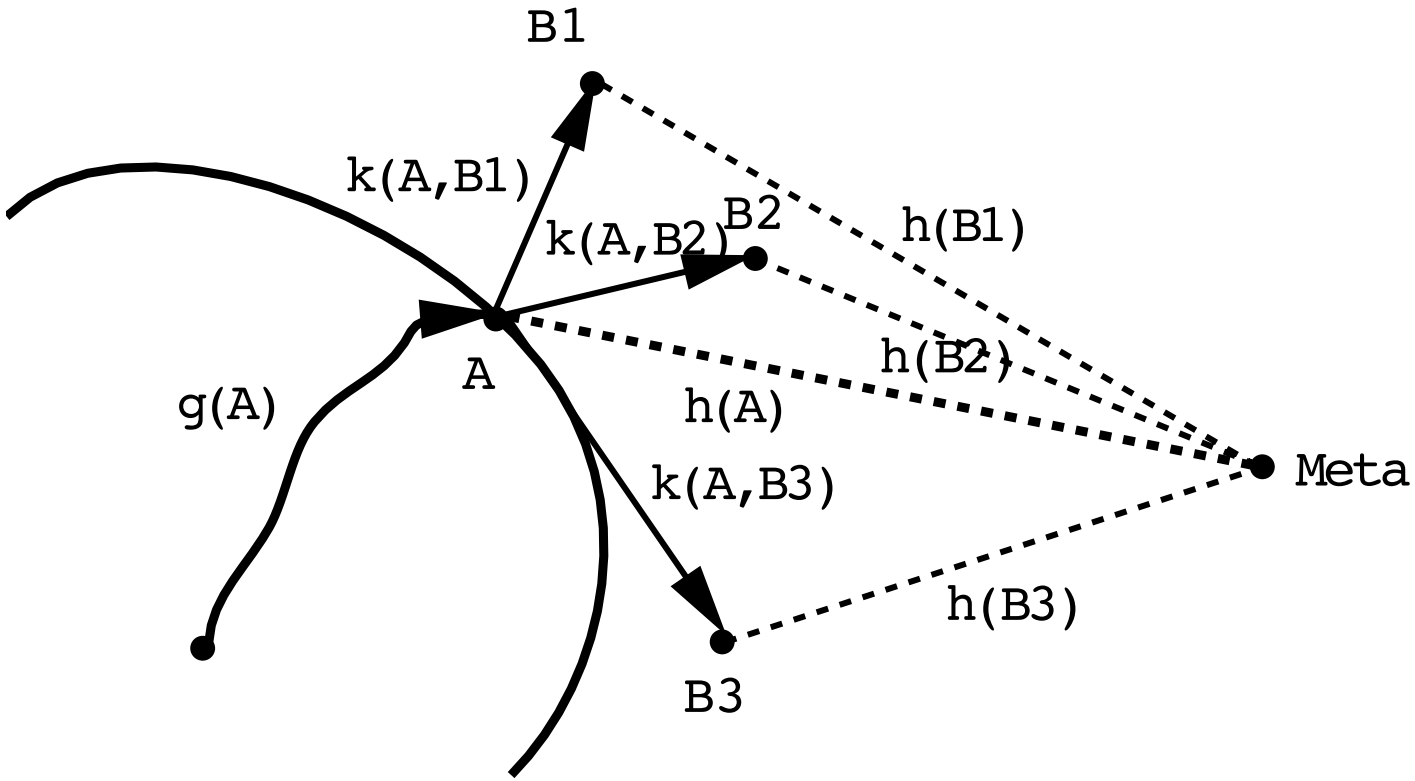


Figure: Esquema ilustrando el método *Back-up*

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Front-to-front method

1. Se calcula alrededor de la meta rutas a varios puntos (B_is),
2. Se toma como nuevo estimador de un nodo A, el mínimo entre las estimaciones de A a cada B_i ($h(A, B_i)$) junto con el costo de cada B_i a la meta ($k_i(B_i, Meta)$), osea: $newh(A) = min_i(k_i(B_i, Meta) + h(A, B_i)))$.
3. De nuevo, si el evaluador no es consistente, entonces se toma:
$newh(A) = max(h(A), min_i(k_i(B_i, Meta) + h(A, B_i)))$.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Front-to-front method

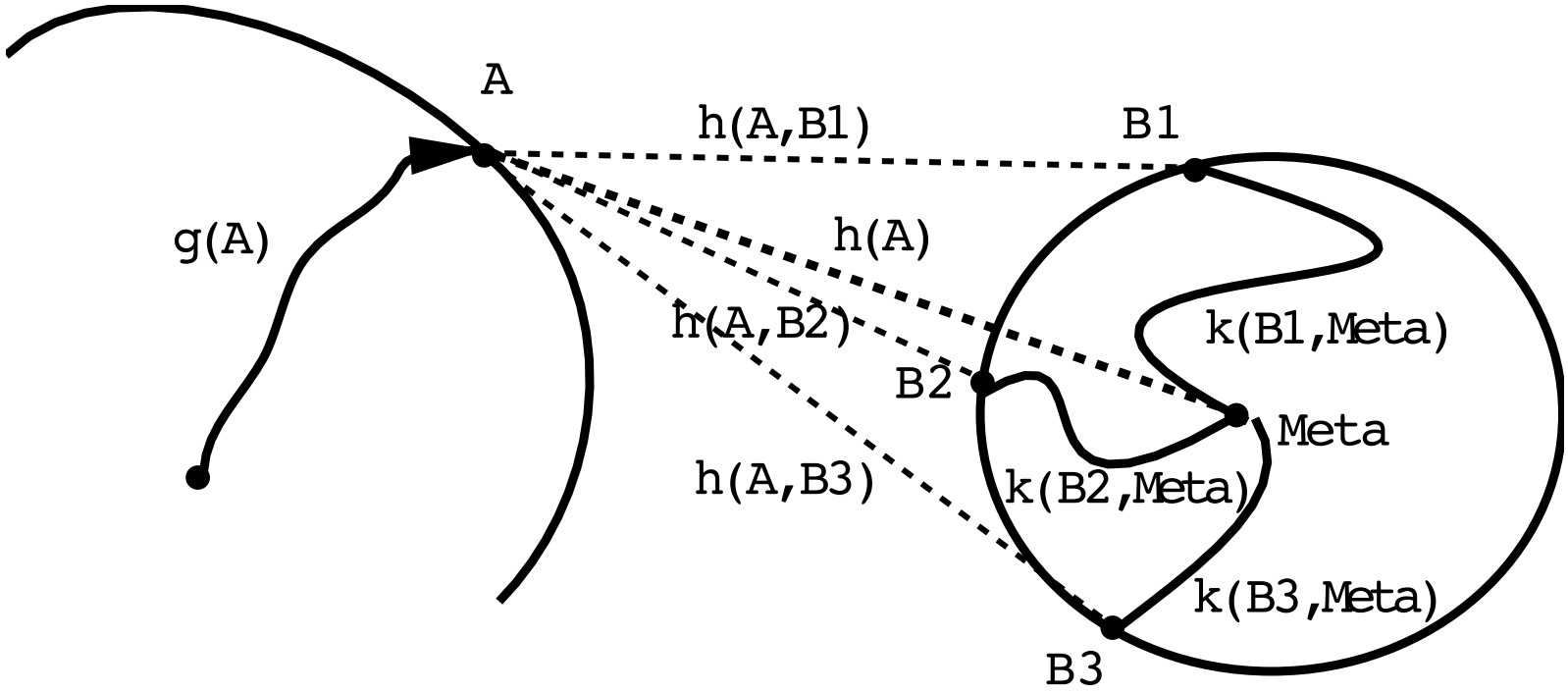
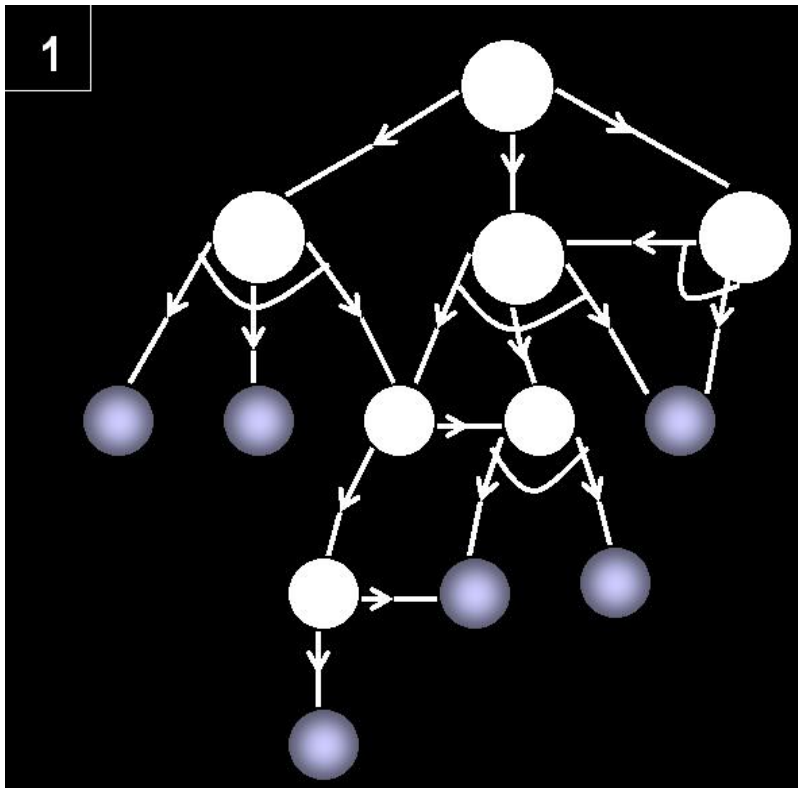


Figure: Esquema ilustrando el método *Front-to-front*

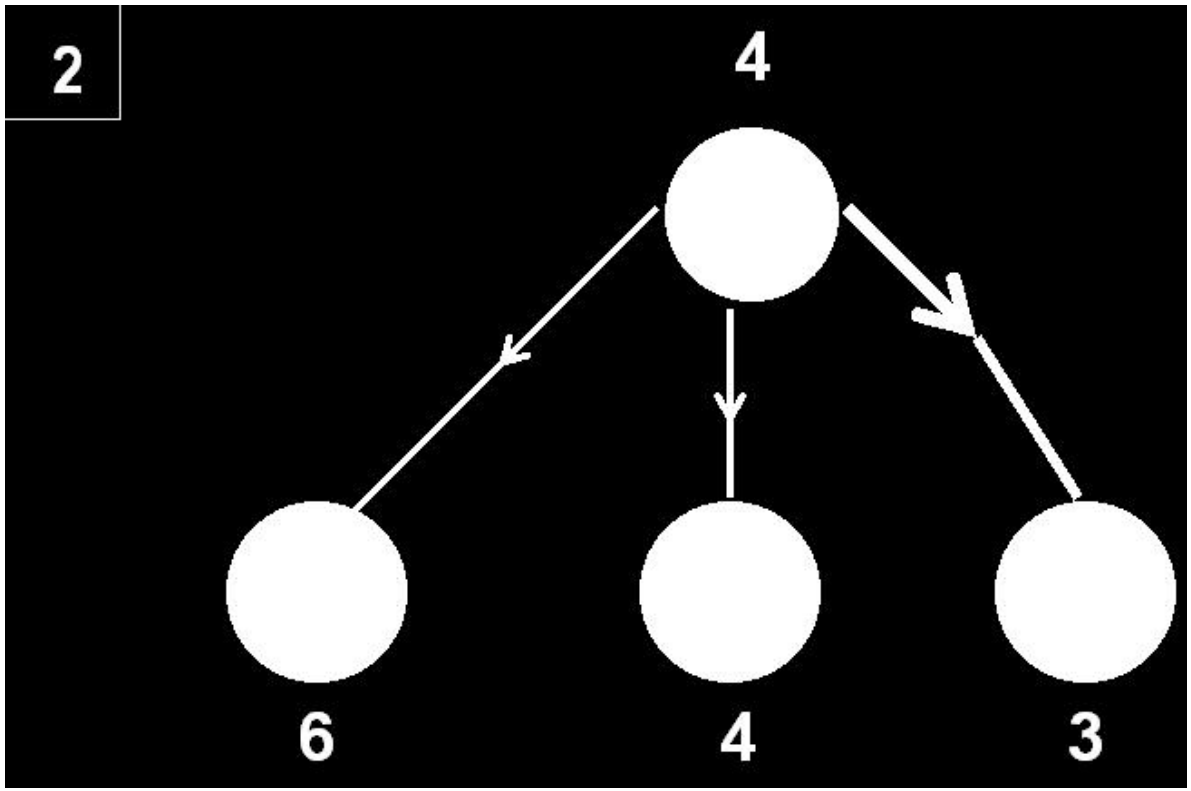
Búsqueda en grafos AND-OR

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS



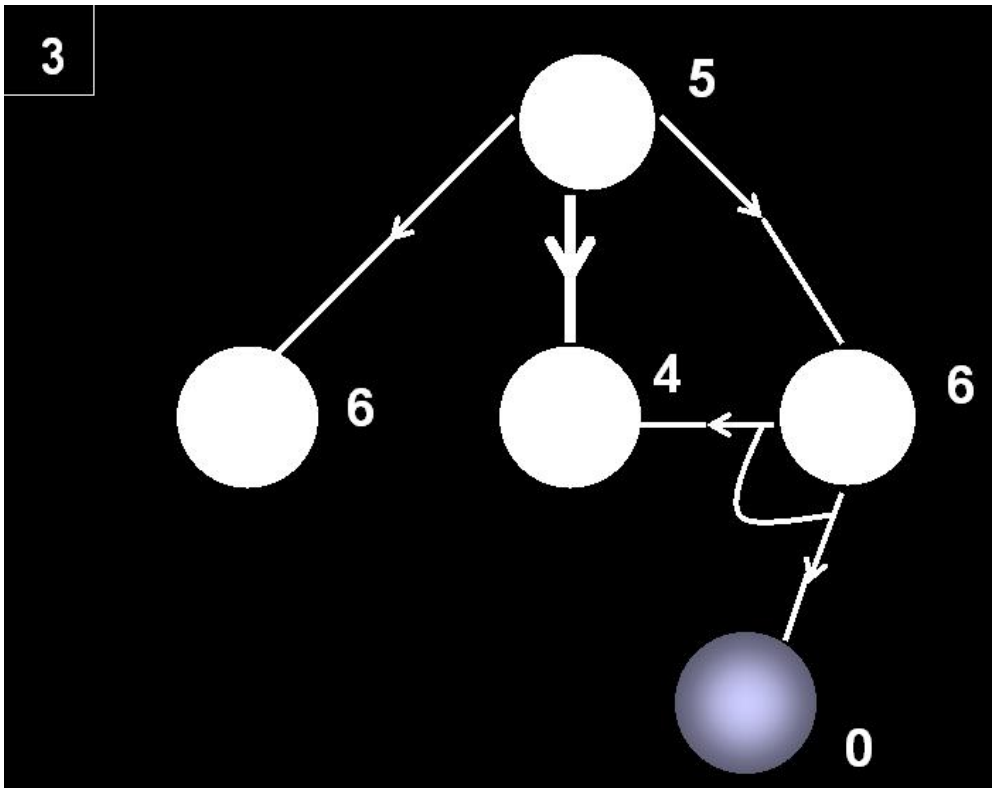
Búsqueda en grafos AND-OR

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS



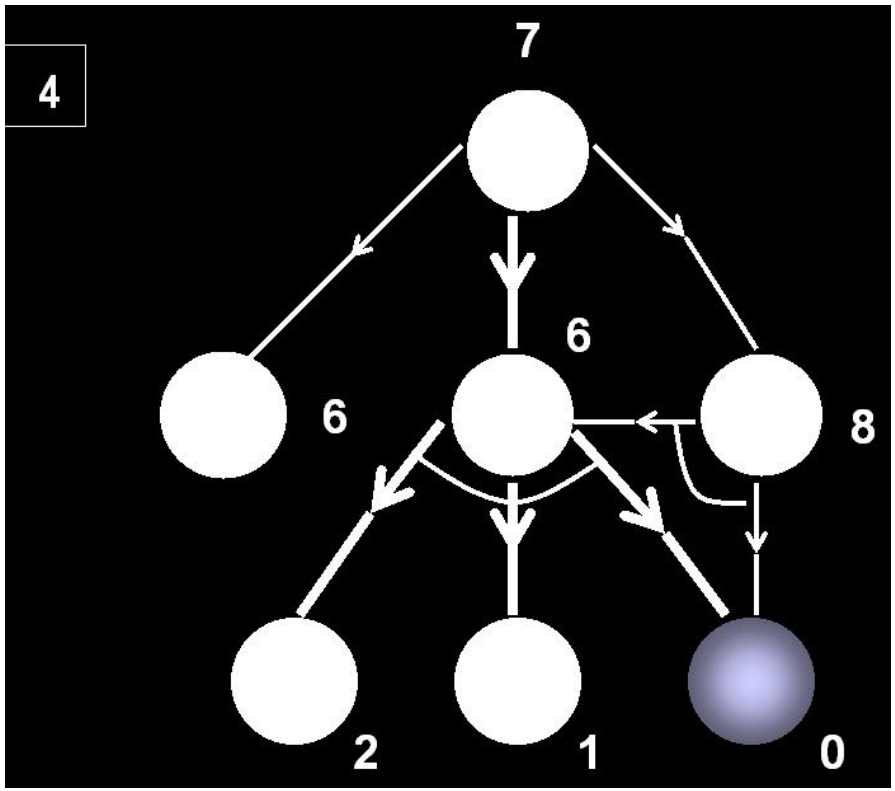
Búsqueda en grafos AND-OR

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS



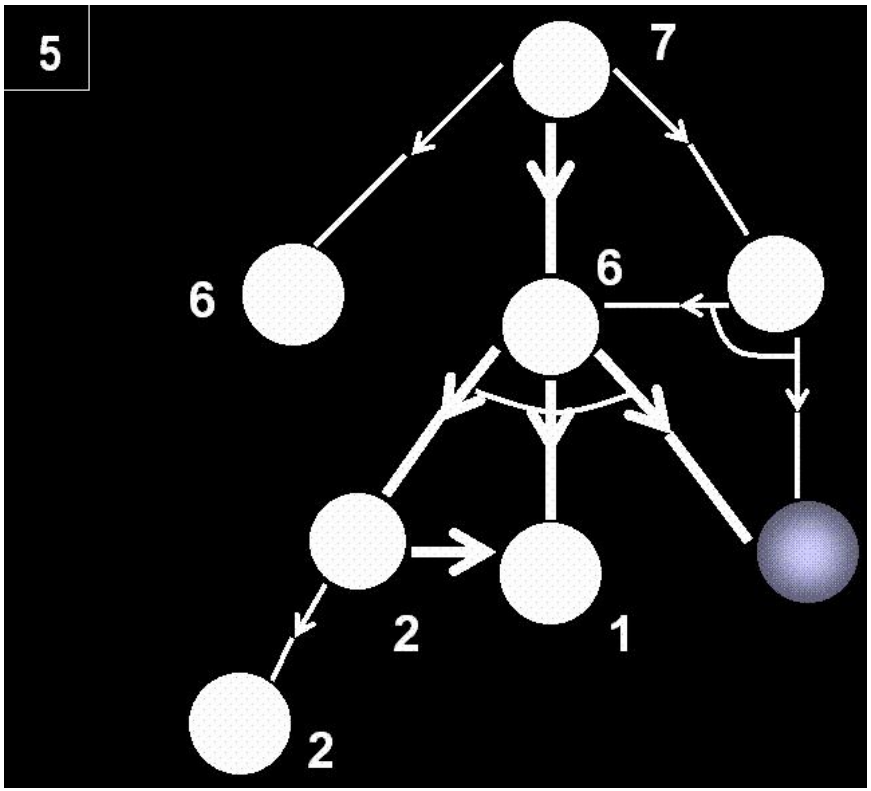
Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Búsqueda en grafos AND-OR



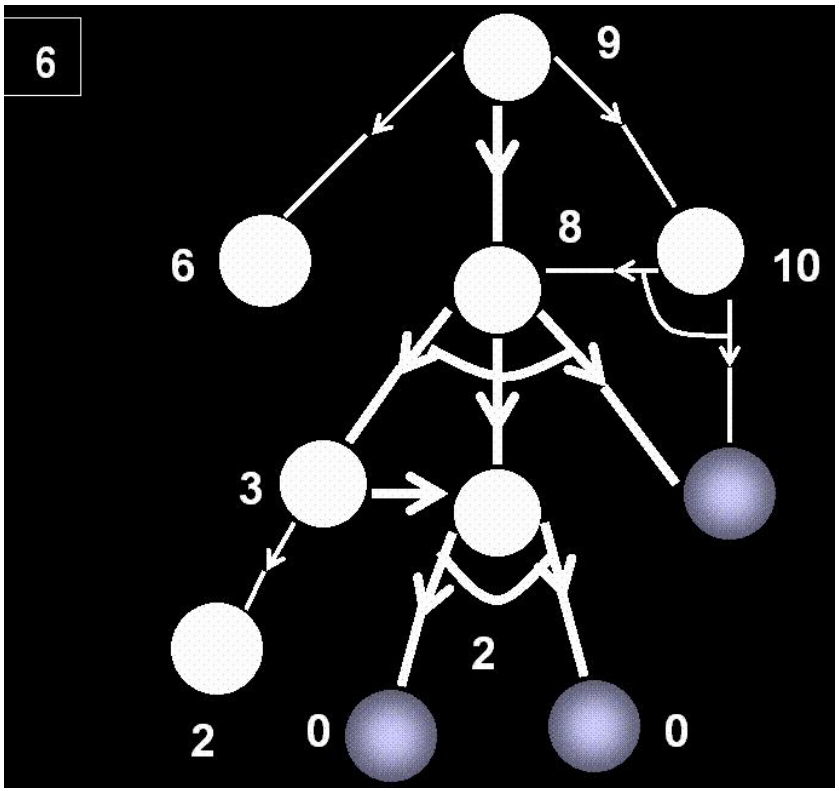
Búsqueda en grafos AND-OR

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS



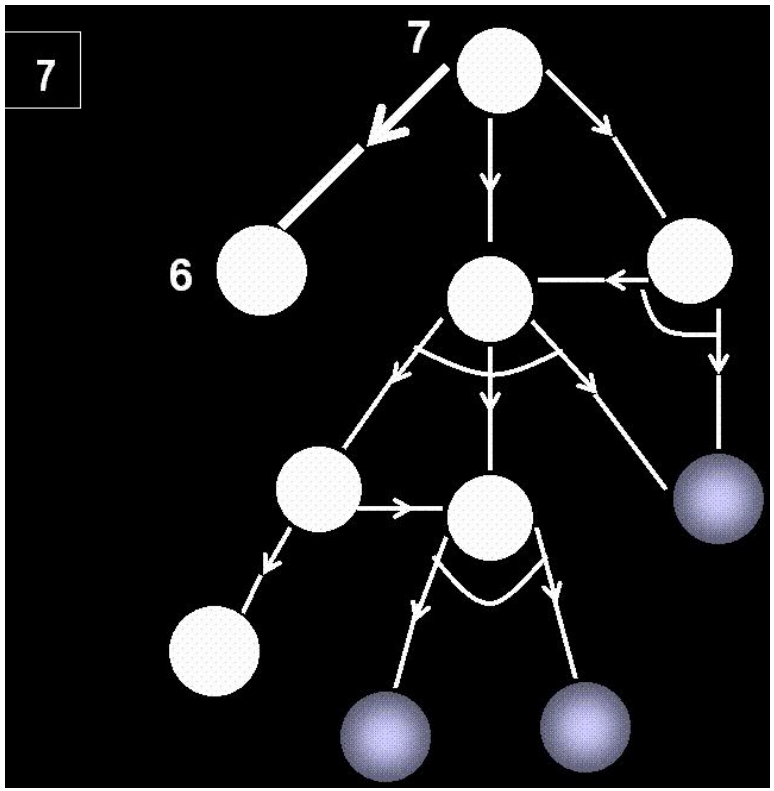
Búsqueda en grafos AND-OR

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS



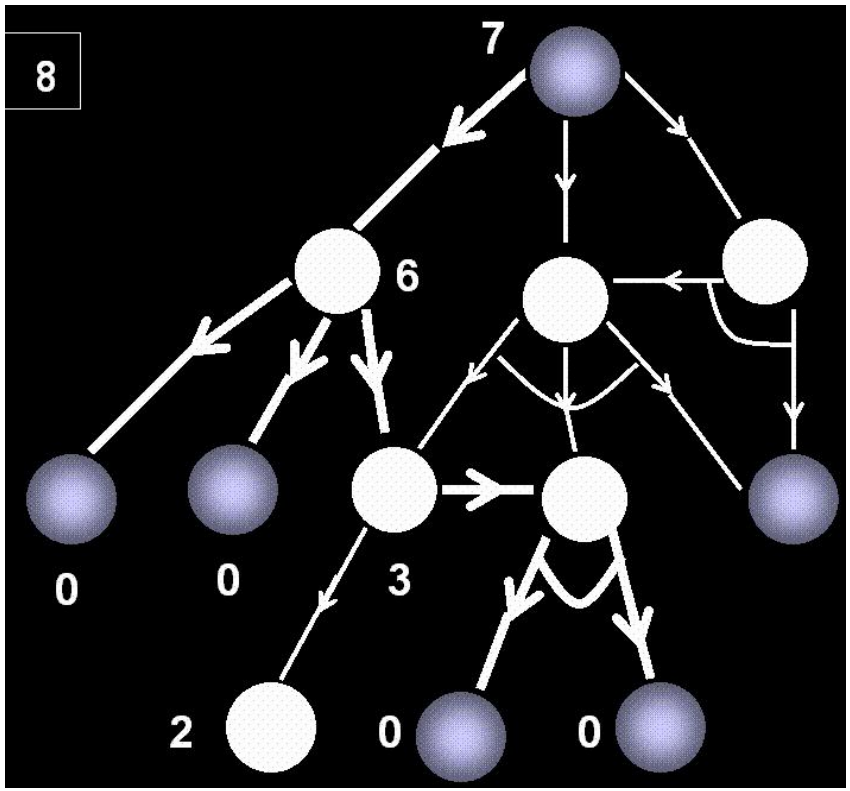
Búsqueda en grafos AND-OR

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS



Búsqueda en grafos AND-OR

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS



Cómo Inventar Heurísticas

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

- Muchas veces el costo de una solución exacta para un problema con menos restricciones o simplificado (*relaxed*) es una buena heurística para el problema original.
- Por ejemplo, podemos usar la notación de STRIPS para representar movimientos

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Cómo Inventar Heurísticas

La accción de mover un cuadro en el 8-puzzle usando notación de Strips.

MUEVE(CuadroX,LugarY,LugarZ):
- Precondiciones:
  - EN(CuadroX,LugarY),
  - LIBRE(CuadroZ),
  - ADY(CuadroY,CuadroZ)
- Añade:
  - EN(CuadroX, LugarZ),
  - LIBRE(LugarY)
- Elimina:
  - EN(CuadroX,LugarY),
  - LIBRE(LugarZ)

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Cómo Inventar Heurísticas

- Si eliminamos las condiciones: LIBRE(CuadroZ), ADY(CuadroY,CuadroZ) nos queda un problema en donde cada cuadro se puede mover a cualquier otro. El número requerido de pasos para resolver este problema es igual al número de cuadros que no están en su lugar (h_1).
- Si sólo eliminamos LIBRE(CuadroZ), corresponde a intercambiar dos cuadros y nos dá la heurística h_2.
- Si eliminamos sólo ADY(CuadroY,CuadroZ), nos dá otra heurística que se introdujo 13 años después de las otras dos.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

Cómo Inventar Heurísticas

- Si queremos añadir más posibilidades, podemos expresar relaciones, tales como *ADJ* en más simples, eg., VECINOS y MISMA–LINEA.
- Eliminando una de las dos, dá nuevas heurísticas.
- Hay que tener cuidado, a veces el problema resultante puede ser más difícil de resolver.
- Si tenemos un conjunto de heurísticas admisibles lo mejor es hacer: $h(n) = max(h_1(n), \ldots, h_m(n))$ con lo cual h domina a todas.
- Otra forma de inventar heurísticas es usando información estadística. Si nuestra heurística nos da un cierto valor diferente podemos usar ese otro valor. Esto deja de garantizar la admisibilidad, pero puede dar en promedio muy buenos resultados.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MACRO–Operadores

- Meta = encontrar una secuencia de operadores que mapea el estado inicial a alguno de los estados finales.
- En muchos casos, algunas de las secuencias de operadores se repiten en problemas diferentes.
- Idea principal: juntar estas secuencias de operadores en macro operadores, “chunks”, etc.
- Considerar a los macros como operadores primarios para aumentar la velocidad de solución en problemas similares.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MACRO–Operadores

Los macros:

- Reducen la profundidad del árbol de búsqueda al considerar secuencias largas de operadores como una sola.
- Aumentan el *branching factor*.
- En el 8-puzzle, las 181,440 posiciones posibles, se pueden resolver sin búsqueda usando una tabla de 35 macros.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MACRO–Operadores

- La tabla de macro operadores para el *8-puzzle* se muestra en la siguiente tabla
- Cada operador es una secuencia de movimientos: up (U), down (D), left (L), right (R) de un cuadro.
- Para usarla se localiza el espacio (cuadro 0) y dependiendo de su posición se realizan los movimientos indicados en la columna de cuadro 0. Luego se sigue con el cuadro 1 y asi sucesivamente.
- Estos macros llegan a la posición final mostrada en la figura 2.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MACRO–Operadores

Macro operadores para el *8-puzzle* (P = posición 0).

P	Cuadros			
	0	1	2	3
0				
1	UL			
2	U	RDLU		
3	UR	DLURRDLU	DLUR	
4	R	LDRURDLU	LDRU	RDLLURDRUL
5	DR	ULDRURD-LDRUL	LURDLDRU	LDRULURDDLUR
6	D	URDLDRUL	ULDDRU	URDDLULDRRUL
7	DL	RULDDRUL	DRUULDRDLU	RULDRDLUL-DRRUL
8	L	DRUL	RULLDDRU	RDLULDRRUL

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MACRO–Operadores

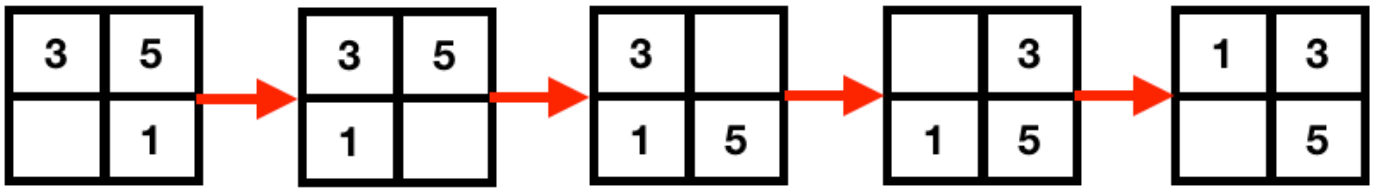
Tabla de macro operadores para el *8-puzzle* (cont.)

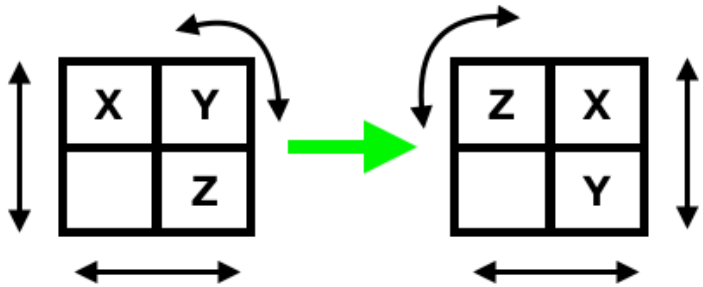
Pos	Cuadros		
	4	5	6
5	LURD		
6	ULDR	RDLLUURDLDRRUL	
7	URDLULDR	ULDRURDLLURD	URDL
8	RULLDR	ULDRRULDLURD	RULD

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MACRO–Operadores

- En el cubo de Rubik ($4 * 10^{19}$) se usan 238 macros.
- Los macros se pueden aprender automáticamente al momento de ir resolviendo un problema
- Estos se incorporan incrementalmente y pueden ayudar a resolver el problema más rápidamente
- Por ejemplo:





Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MACRO–Operadores

- El macro anterior lo podemos aplicar, a partir de su creación, en cualquier situación en donde se pueda aplicar
- La descomposición de problemas en sub–metas juega un papel fundamental en la creación de macros.
- El beneficio de los macros decae al incrementar su número (i.e., al resolver más problemas) y es necesario usar filtros de *aplicabilidad*.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MACRO–Operadores

- Al aplicar los filtros hay que considerar que los macros no son unidades totalmente independientes.
- Se deben de considerar cuándo y bajo qué condiciones crear/eliminar macros
- En general los macros aprendidos y su filtrado dependen de la función de evaluación (heurística) que se usa durante la búsqueda de la solución del problema.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MACRO–Operadores

En general:

- Una buena función de evaluación requiere de un filtro *agresivo*.
- Con una mala función de evaluación los macros son indispensables y el filtro puede ser más *laxo*.
- Con una muy buena función de evaluación los macros no sirven.
- Se pueden utilizar en aprendizaje incremental.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

MCTS (*Monte Carlo Tree Search*)

- Es un método para encontrar decisiones óptimas tomando muestras aleatorias y construyendo un árbol de búsqueda de acuerdo a los resultados
- Desató recientemente un gran interes dados los resultados que se empezaron a obtener en el juego de Go
- Idea: analizar los movimientos más promisorios expandiendo el árbol de búsqueda haciendo muestreo aleatorio

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

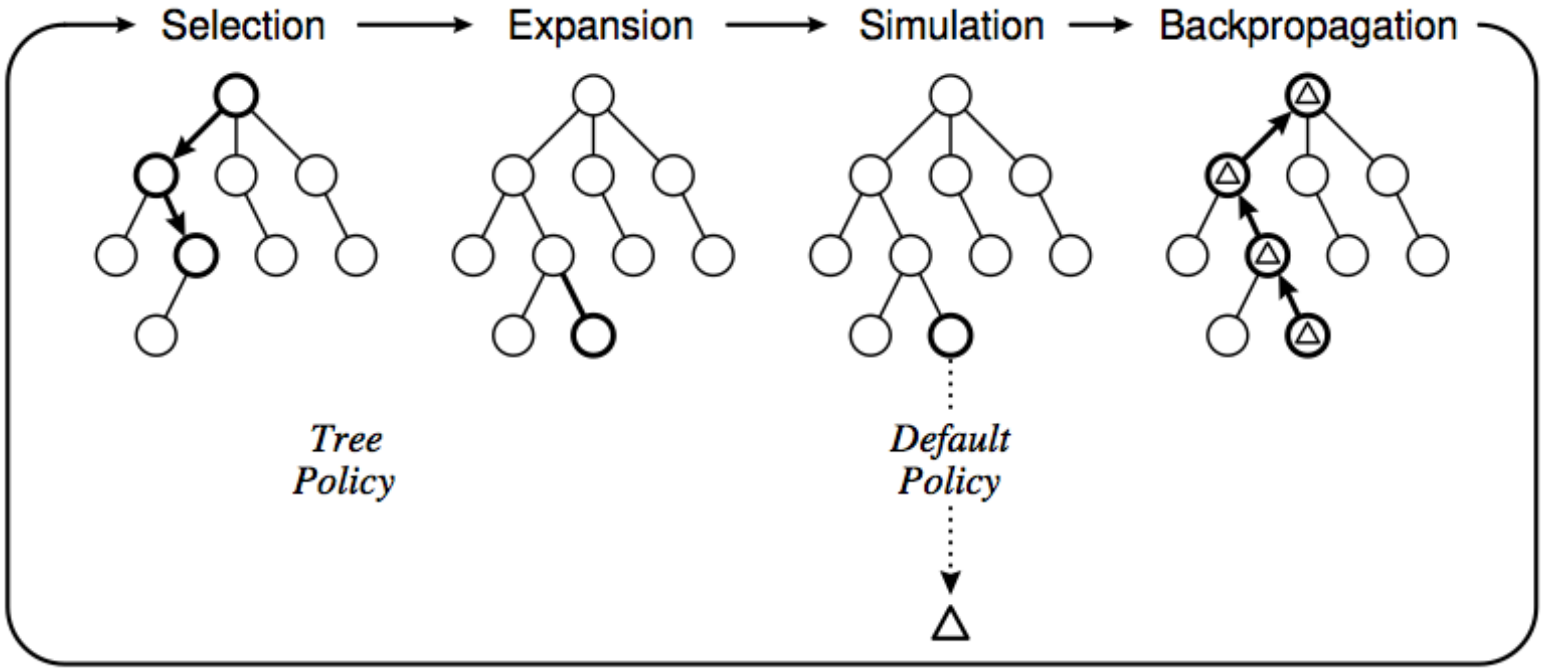
MCTS

Cada ronda de MCTS consiste de 4 pasos:

1. Selección: empezando de la raíz se seleccionan hijos sucesores hasta llegar a una hoja (cómo seleccionar ver abajo)
2. Expansión: a menos que la hoja termine el juego, crea sus hijos y seleciona uno de ellos
3. Simulación: Empieza un juego aleatorio de ese nodo (*random playout*): En un *playout* el juego se lleva hasta el final seleccionado acciones aleatorias
4. Retropropagación: usando la información del *playout* se actualiza la información en los nodos del camino del nodo raíz al nodo desde empezó la simulación

MCTS

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS



Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

UCT (*Upper Confidence Bound applied to Trees*)

- Un aspecto fundamental de los MCTS es cómo balancear exploración y explotación.
- Osea explotación de variantes profundas después de movimientos con promedios altos y la exploración de movimientos con pocas simulaciones
- Un muestreo es selectivo puede hacer grandes ahorros de cómputo
- La idea de UCT es aplicar ideas de problemas *multi-armed bandit* (UCB1) para guiar como explorar/explotar ese árbol

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

UCT

La fórmula básica de UCT es seleccionar en cada nodo del árbol, la movida para la cual la expresión siguiente tenga el valor máximo:

$$\text{argmax}_{v' \in hijos(v)} \frac{Q(v')}{N(v')} + c\sqrt{\frac{\ln N(v)}{N(v')}}$$

donde:

- $Q(v')$: recompensas totales (el número de veces que se gana después del i-ésimo movimiento)
- $N(v')$: el número de veces que se ha visitado (simulaciones) el i-ésimo movimiento
- $N(v)$: el número total de simulaciones que se han hecho en el nodo (igual a la suma de todas las v's)
- c: es el parámetro de exploración, en teoría igual a $\sqrt{2}$, pero en la práctica seleccionado empíricamente

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

UCT

Ventajas:

- No requiere la definición de una función de evaluación (no siempre es fácil definir una)
- El árbol de juego crece asimétricamente, ya que se concentra en las partes más promisorias, con lo que le va bien en juegos con factores de arborecencia grandes
- MCTS se puede parar en cualquier momento (*any-time*)

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

UCT

- Los *playouts* pueden ser “light” (usando movidas aleatorias) o “heavy” (usando heurísticas para seleccionar las movidas)
- Para las estadísticas se pueden aprovechar pedazos de juegos que se repitan.
- Por ejemplo en GO algunas jugadas/posiciones se repiten varias veces en el juego y están relativamente aisladas de las demás, por lo que se pueden aprovechar todas las estadísticas de éstas.

Introducción
Ejemplos e Ideas
Procedimientos de Búsqueda Clásicos
Búsqueda ciega o sin información
Búsqueda con Información
Cómo Inventar Heurísticas
MACRO–Operadores
MCTS

UCT

Existen varias versiones paralelas:

- Paralelización de hojas: executar varios *playouts* en paralelo
- Paralelización de raíz: construir varios árboles en paralelo y realizar el movimiento basado en las ramas de nodo raíz de todos los árboles
- Paralelización del árbol: construir en paralelo el árbol de juego cuidando de posibles conflictos