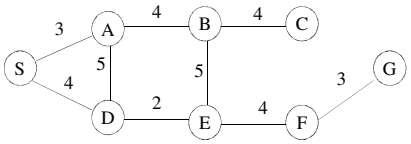
Búsqueda

Introducción

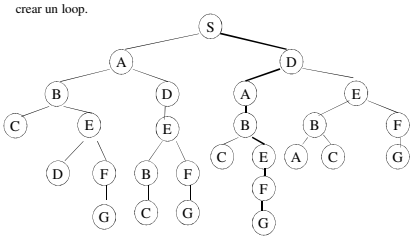
- Problema
  - Suponga que se quiere descubrir el camino de una ciudad (S) a otra (G) usando un mapa



- Para encontrar el mejor camino dos costos diferentes deben ser considerados
  - Costo computacional gasto para encontrar un camino
  - Costo del viaje por utilizar este camino
- Posibles soluciones
  - Viajes frecuentes, vale la pena gastar tiempo viajando hasta encontrar un buen camino
  - Viaje esporádico y difícil de encontrar un camino: basta con encontrar un camino

Introducción

- El problema puede ser representado por una red (grafo)
  - Al recorrer la red el mismo nodo debe ser visitado mas de una vez
    - Representa un ciclo lo que quiere decir que pueden aparecer loops infinitos
    - Solución remover loops
- Remover loops
  - Solucion extender todos los caminos posibles hasta no poder extender mas ninguno sin crear un loop.

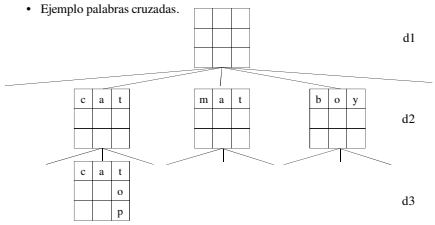


Introducción

- Un sistema de IA puede resolver problemas de la misma forma
  - El sistema sabe donde esta (conjunto de informaciones iniciales)
  - El sabe donde desea ir (Estado Objetivo)
- Resolver problemas en IA implica búsqueda del estado objetivo (paradigma de solución de problemas)
  - Forma simplificada de raciocinio
  - Simples problemas de IA reducen el raciocionio a búsqueda
- Problemas de búsqueda son frecuentemente descritos utilizando diagramas de árboles de búsqueda
  - Árboles semánticas donde cada nodo denota un paso en el camino del nodo inicial para el nodo objetivo
    - Nodo Inicial (I) = donde la búsqueda comienza
    - Nodo Objetivo(O) = donde ella termina
  - El Objetivo es encontrar un camino que una el nodo inicial con el nodo objetivo.

Introducción

- Problema de búsqueda
  - Entrada:
    - Descripción de los nodos iniciales y el objetivo
    - Procedimiento que produce los siguiente de un nodo dado
  - Salida:
    - Secuencia valida de nodos, iniciando con el nodo inicial terminando con el nodo objetivo
    - Ejemplo palabras cruzadas.



Introducción

- Definiciones importantes
  - Profundidad: número de uniones entre un nodo dado y la raíz
  - Amplitud: número de sucesores de un nodo hijo
  - Nodo Raíz: Nodo que no tiene padre
  - Nodo Hoja: Nodo que no tiene hijos descendientes
  - Nodo Objetivo: Nodo que satisface el problema
- Definiciones importantes
  - Camino Parcial: Camino final donde el nodo final es un nodo hoja y no el objetivo
  - Camino Final o Completo: Camino donde el nodo final es un nodo objetivo
  - Expandir un Nodo: Crear los hijos de un nodo
  - En búsqueda se aprende como encontrar un camino entre el nodo inicial y un nodo objetivo
- Problemas de la operación de búsqueda
  - Con el aumento del tamaño del árbol de búsqueda y el número de posibles caminos el tiempo de búsqueda aumenta
  - Existen varias formas de reducir el tiempo de búsqueda
- Posibles situaciones
  - Mas de un nodo objetivo
  - Mas de un nodo inicial
    - En estas situaciones encontrar cualquier camino de un nodo inicial para un nodo objetivo
    - Encontrar el mejor camino

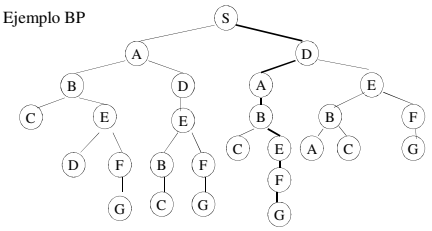
Introducción

- Existen varios tipos de algoritmos de búsqueda
  - Lo que los distingue es la manera de elegir los nodos
- Métodos de Búsqueda
  - Búsqueda ciega: la elección depende de la posición del nodo en la lista (escojo el primer elemento)
  - Búsqueda heurística: Escoge usando informaciones especificas del dominio para ayudar en la decisión de elegir el nodo
- Búsqueda ciega
  - Manera mas directa de encontrar una solución
    - Visitar todos los caminos posibles sin repetir el mismo nodo.
    - No usa informaciones para guiar la búsqueda
    - La estrategia aplicada es una búsqueda exhaustiva hasta encontrar una solución o fallar
  - Ejemplo encontrar el camino para ir a Lima
    - Punto de Partida Arequipa
    - Objetivo Lima
    - Usar un mapa de carreteras

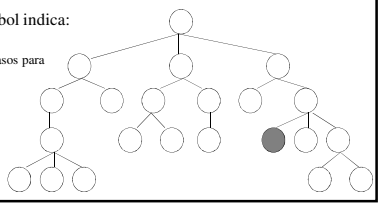
Búsqueda Ciega

- Existe un gran numero de técnicas
  - Búsqueda en profundidad (BP)
    - El árbol es examinado de arriba para abajo
    - Aconsejable donde los caminos improductivos no son muy largos
  - Búsqueda en amplitud.
    - El árbol es examinado de izquierda a derecha
    - Aconsejable cuando el numero de ramos (hijos) de los nodos no son muy grandes (no vale la pena cunado los nodos objetivos están en el mismo nivel)
- Algoritmo
  1. Definir un conjunto de L de nodos iniciales
  2. Si L es vació
     Entonces la búsqueda no tubo solución
     Si no entonces n es el primer nodo de L
  3. Si n es un nodo objetivo
     Entonces retornar el camino del nodo inicial hasta n
     Parar
     Sino Remover n de L
     Adicionar al inicio todos Hijos de n, etiquetando cada de ellos con el camino para el nodo inicial, volver al paso2

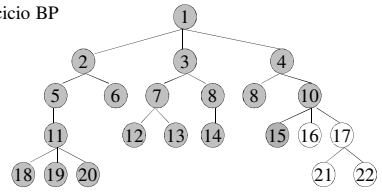
Ejemplo BP



- Dado el siguiente árbol indica:
  - La memoria máxima
  - Número mínimo de pasos para conseguir el objetivo



Ejercicio BP



- 1 L = {1}
- 2 L = {21, 31,41}
- 3 L = {52-1,62-1,31,41}
- 4 L = {115-2-1, 62-1,31,41}
- 5 L = {1811-5-2-1, 1911-5-2-1, 2011-5-2-1,62-1,31,41}
- 6 L = {1911-5-2-1, 2011-5-2-1 62-1,31,41}
- 7 L = {2011-5-2-1, 62-1,31,41}
- 8 L = {62-1,31,41}
- 9 L = {31,41}

....

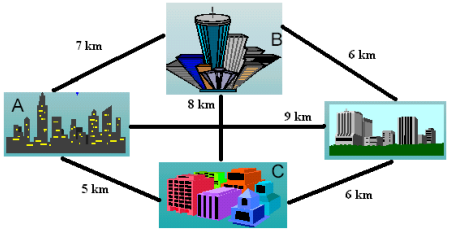
- 18 L = {1510-4-1,1610-4-1,1710-4-1}

Memoria máxima = 6
Nro de pasos = 18

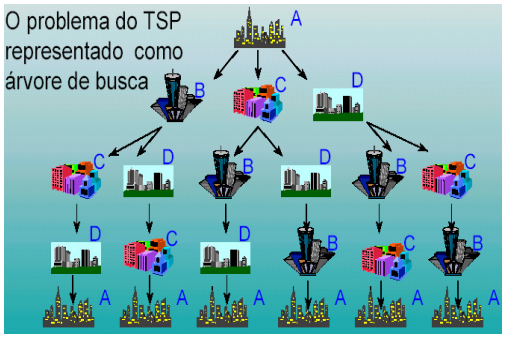
- Algoritmo BA
  1. Definir un conjunto de L de nodos iniciales
  2. Si L es vació
     Entonces la búsqueda no tubo solución
     Si no entonces n es el primer nodo de L
  3. Si n es un nodo objetivo
     Entonces retornar el camino del nodo inicial hasta n
     Parar
     Sino Remover n de L
     Adicionar al final de L todos los hijos de n, etiquetando cada de ellos con el camino para el nodo inicial,
     Volver al paso2
- Búsqueda no Deterministica
  - Escoge aleatoriamente el nodo del árbol a ser expandida
    - Tiro de suerte
  - Posiblemente mas ventajosa para árboles pequeños, con unos pocos ramos
    - Alternativa valida solo si BP y BA son impracticables
- Algoritmo BN
  1. Definir un conjunto de L de nodos iniciales
  2. Si L es vació
     Entonces la búsqueda no tubo solución
     Si no entonces n es el primer nodo de L
  3. Si n es un nodo objetivo
     Entonces retornar el camino del nodo inicial hasta n
     Parar
     Sino Remover n de L
     Adicionar en posiciones aleatorias de L todos los hijos de n, etiquetando cada de ellos con el camino para el nodo inicial,
     Volver al paso2

Ejemplos de Búsqueda

- Problema del cajero viajero (TSP)
  - Un cajero viajante debe visitar N ciudades en su área de ventas
  - El cajero comienza de una base visita cada ciudad una sola vez y retorna a su ciudad al final
  - A cada viaje esta asociado un costo proporcional a la distancia recorrida
  - El cajero debe recorrer la ruta mas corta



Problema del TSP



Problema del TSP

- Problema es una explosión combinatoria
  - Con cuatro ciudades existe 6 caminos posibles
  - Con 10 existen 386.880 caminos posibles
  - Cuanto mas ciudades fueran aumentadas al TSP mas caminos posibles van a existir
  - Lo que lleva a una explosión combinatoria
  - Como prevenir o por lo menos como evitar eso?
- Búsqueda ciega nos es eficiente
  - Es necesario limitar de alguna forma el espacio de búsqueda para tornarlo mas rápida y eficiente
  - La búsqueda seria mas eficiente si las elecciones fuesen ordenadas
    - Las posibilidades mas provisorias serian exploradas antes
    - En varias alternativas es posible determinar un ordenamiento razonable
    - Alternativas pueden ser ordenadas a través de heurísticas
    - Por ejemplo si desea ir a su casa puede tomar ciertas consideraciones
      - Si vive para norte ignora los precios de los taxis para la zona sur
      - Si vive para el sur ignora el precio para el norte
      - Estas heurísticas nos ayudan a limitar la búsqueda

Búsqueda

- Los humano usa ciertas consideraciones
  - En IA estos consejos son llamadas de heurísticas
  - Búsquedas heurísticas
- Métodos de búsqueda heurística
  - HillClimbing
  - El mejor el primero
- Busqueda Heuristica
  - Observación
    - Tiempo de gasto evaluando la heurística debe ser recupero por una reducción en el espacio de búsqueda
    - Actividad a nivel base: esfuerzo gastado intentando resolver el problema
    - Actividad a nivel meta: trabajo gastado en como resolver el problema
    - Existe un trade-off en dichas actividades
      - Búsqueda eficiente: es el tiempo gastado en el nivel meta y recuperado con reducciones en el tiempo necesario para el nivel base
      - A veces es mejor definir un nuevo espacio de búsqueda

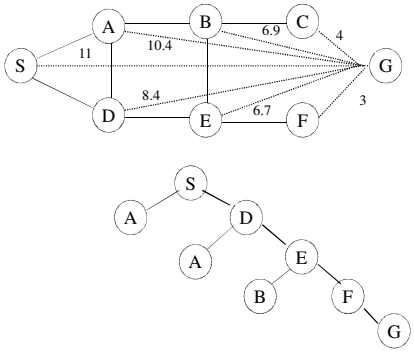
HillClimbing

- Funcionamiento
  - Buscar entre los nodos próximos el mas cercano al objetivo
    - Seleccionar el hijo mas cercano del objetivo
    - Radio de visión limitado a la proximidad del nodo actual
  - Semejante a optimización de una función
    - Buscar la combinación de valores de los parámetros que hacen que la función asuma el mayor valor
- Ejemplo de funcionamiento
  - Imagine que quiera escalar una montaña y:
    - Esta haciendo una neblina fuerte
    - Posee apenas una brújula y un altímetro
    - Buscar el punto mas alto de un terreno durante una caminata
  - Posible solución dar un paso en cada dirección y escoger aquella que Ud mas crea por conveniente
- Características
  - Funcionamiento como una BP pero escoge el hijo en función de la distancia al objetivo
  - Cuanto mejor la medida heurística mas eficiente es la búsqueda
  - Cantidad mayor de conocimiento lleva a una reducción en el tiempo de búsqueda
  - Ejemplo: la medida como de una distancia física al objetivo

Algoritmo Hill Climbing

1. Definir un conjunto de L de nodos iniciales clasificados de acuerdo con las distancias al objetivo (en orden creciente)
2. Si L es vació
   Entonces la búsqueda no tubo solución
   Si no entonces n es el primer nodo de L
3. Si n es un nodo objetivo
   Entonces retornar el camino del nodo inicial hasta n
   Parar
   Sino Remover n de L
   Ordenar los hijos de n en orden creciente de a cuerdo con las distancias al nodo objetivo
   Adicionar al inicio de L todos los hijos de n, etiquetando cada de ellos con el camino para el nodo inicial,
   Volver al paso2

Ejemplo Hill Climbing



Hill Climbing

- Problemas
  - Menor camino de una ciudad para otra puede llevar a otra mas distante
    - Opción 1 volver atrás y tomar el segundo menor camino pero esto lleva tiempo
    - Opción 2 incluir no determinismo
    - Opción 3 usar otras heurísticas
  - Máximo local: existe una solución la cual puede no ser la mas adecuada
  - Planicie: todos los vecinos llevan para el mismo valor
  - Arista: existe por lo menos una dirección que aumenta el valor pero ninguna transición sigue ese camino

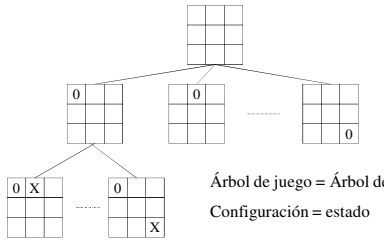
Búsqueda Mejor el Primero

- Funcionamiento
  - Sigue por el mejor nodo abierto (que aun tiene hijo para ser visitado)
  - Es como el Hil Climbing sin la restricción de búsqueda en profundidad
    - Escoge el mejor de la lista L
    - Generalmente encuentra caminos mas cortos que el Hill Climbing
    - Siempre mueve en dirección del nodo mas próximo del objetivo no importa donde el nodo puede estar en el árbol
- Algoritmo

1. Definir un conjunto de L de nodos iniciales
2. Sea n el nodo mas próximo del objetivo
   si L es vació
   Entonces la búsqueda no tubo solución
3. Si n es un nodo objetivo
   Entonces retornar el camino del nodo inicial hasta n
   Parar
   Sino Remover n de L
   Adicionar a L todos los hijos de n, etiquetando cada de ellos con el camino para el nodo inicial,
   Volver al paso2

Juegos y Búsqueda Competitiva

- Árboles de juego: árbol semántico en el cual los nodos representan configuraciones del tablero y las transiciones del juego



Árbol de juego = Árbol de estados

Configuración = estado

Algoritmo A*

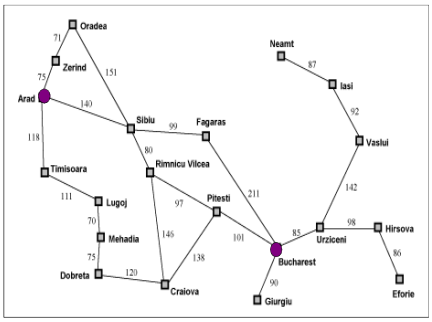
- A* expande el nodo de menor valor de *f* en la frontera del espacio de estados
- Intenta minimizar el costo total de la solución combinando:
  - Busqueda Golosa (h)
    - Mejor el primero
  - Busqueda de Costo Uniforme (g)
    - Ineficiente, pero completa
- *f* - Función de evaluación del A*:
  - $f(n) = g(n) + h(n)$
  - $g(n)$ = distancia de n al nodo inicial
  - $h(n)$ = distancia estimada de n al nodo final

Algoritmo A*

- Se *h* es *admisible*, entonces $f(n)$ es admisible tambien
  - Es decir *f* nunca irá superestimar el costo real de la mejor solución através de *n*
  - pues *g* guarda el valor exacto del camino ya recorrido.
- Con A*, la ruta elegida entre *ciudades* es el hecho de mas corta.

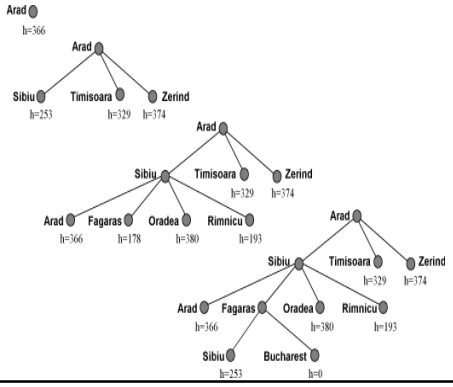
Algoritmo A*: ejemplo

Viajar de Arad a Bucharest

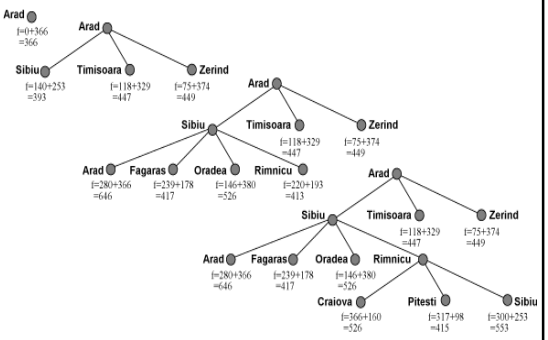


Straight line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Por la Busqueda del Mejor Explorado



Usando A*



Algoritmo A*:
Análisis del comportamiento

- La estratégia es completa y optima
- Costo de tiempo:
  - Exponencial con la longitud de la solución, por lo tanto buenas heristicas disminuyen ese costo significativamente
    - El factor de expansión queda proximo de 1
- Costo memória:
  - Guarda todos los nodos expandidos en la memoria, para posibilitar el *backtracking*

Algoritmo A*
Análise do comportamento

- La estratégia presenta eficiencia optima
  - ningun otro algoritmo optimo garantiza expandir menos nodos
- A* solo expande nodos con $f(n) \leq C^*$, donde C^* *es* el costo del camino optimo
- Para garantizar la optimalidad del A*,el valor de f en un camino particular deve ser no decresciente!!!
  - $f(sucessor(n)) \geq f(n)$
  - i.e., el costo de cada nodo generado en el mismo camino nunca es menor que el costo de sus antecesores

Algoritmo A*
Analisis del comportamiento

- $f = g + h$ debe ser no decresciente
  - g es no decresciente (para operadores no negativos)
    - costo real del camino ya recorrido
  - h debe ser no-cresciente (consistente, monotonico)
    - $h(n) \geq h(sucesor(n))$
    - i.e., cuanto mas próximo del nodo final, menor el valor de h
    - eso vale para a mayoria de las funciones heurísticas
- Cuando h no es consistente, para garantirzar optimidad del A*, tenemos:
  - Cuando $f(suc(n)) < f(n)$
  - Se usa $f(suc(n)) = \max(f(n), g(suc(n)) + h(suc(n)))$