Solución de Problemas Mediante Búsqueda (1)

Carlos Hurtado L.
Depto de Ciencias de la Computación, Universidad de Chile

Contenido

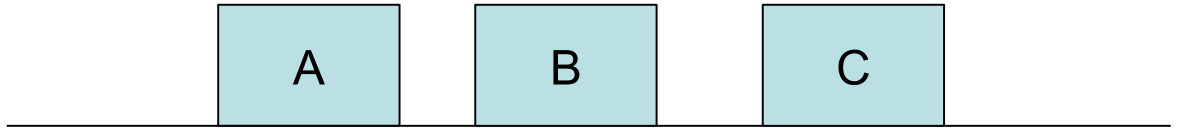
- Solución de problemas mediante búsqueda
  - Modelación de problemas como búsquedas
  - Estrategias de búsquedas (heurísticas)
  - Juegos
- Referencias: capítulos 7, 8 y 9. Inteligencia Artificial, Una Nueva Síntesis. N. Nilsson

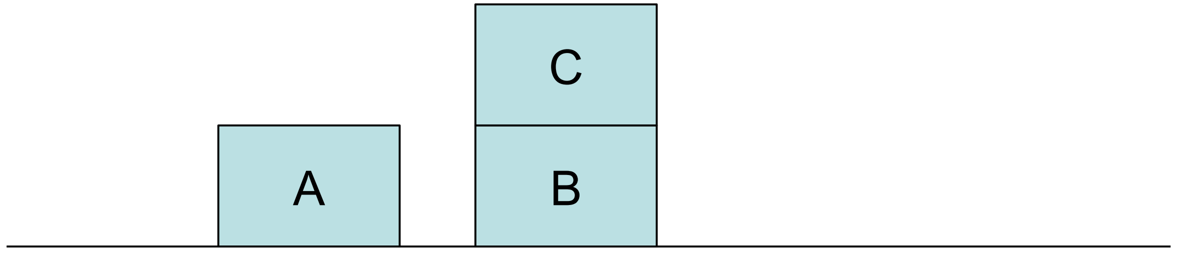
Problemas de Planificación

- Dado un “agente” que opera en un “mundo” a través de “acciones”
  - Las acciones producen cambios en el mundo
- Dado un objetivo
  - estado del mundo al que se quiere llegar
- Encontrar una secuencia de acciones que debe realizar el agente para cumplir el objetivo

Ejemplo

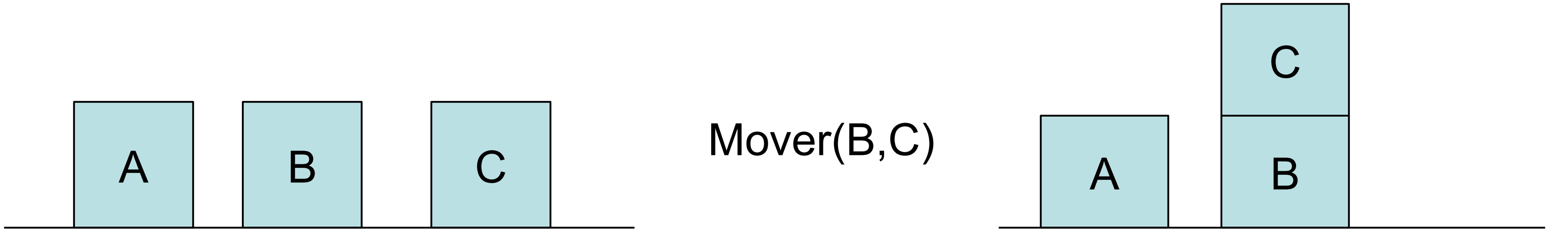
- Mundo de bloques





Mundo de Bloques

- Acciones del Agente:
  - mover(x,y): pone el bloque X sobre Y
  - X es A,B, o C; Y es A,B,C o “suelo”.



Grafo de Estados

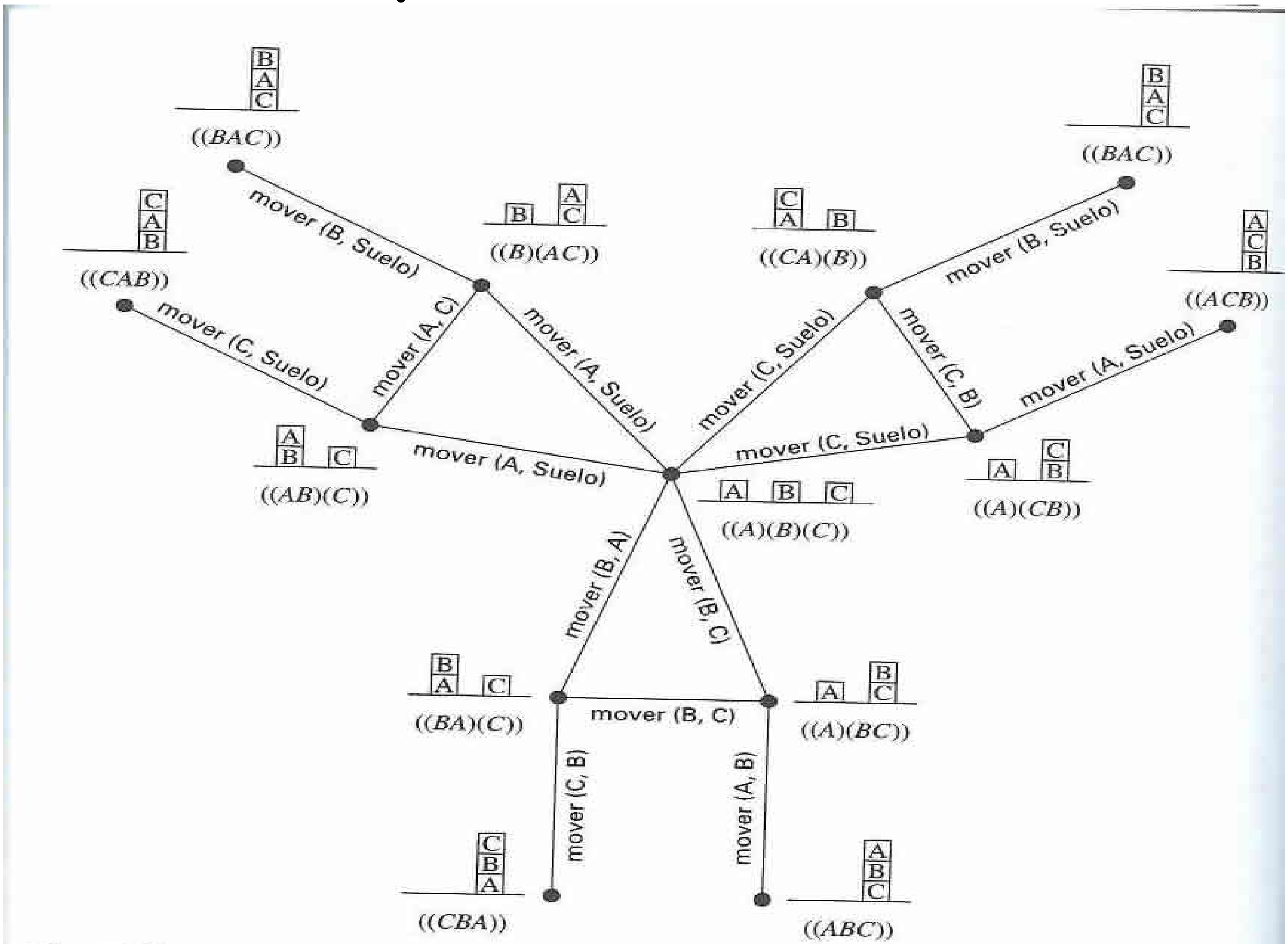


Figura 7.2.

Planificiación y Búsqueda en Grafos

- El “plan” del agente puede ser visto como un camino en el grafo de estados
  - Esto suponiendo que el grafo es estático (el mundo sólo cambia por efecto de las acciones del agente)
  - En caso contrario encontrar un plan es más complejo
- Planificación: encontrar un camino entre el estado inicial y el estado objetivo
- Podemos estar interesados en determinados caminos:
  - El más corto, el de menor costo, sujeto a restricciones, etc.

Ejemplo: puzzle de ocho piezas

Estado Inicial

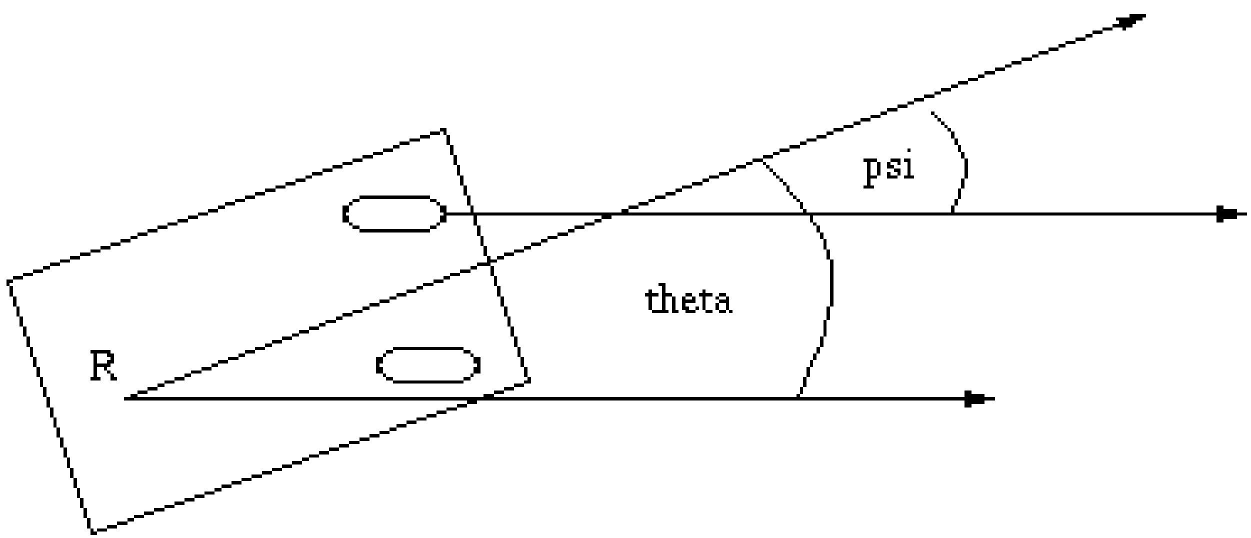
2	8	3
1	6	4
7		5

Objetivo

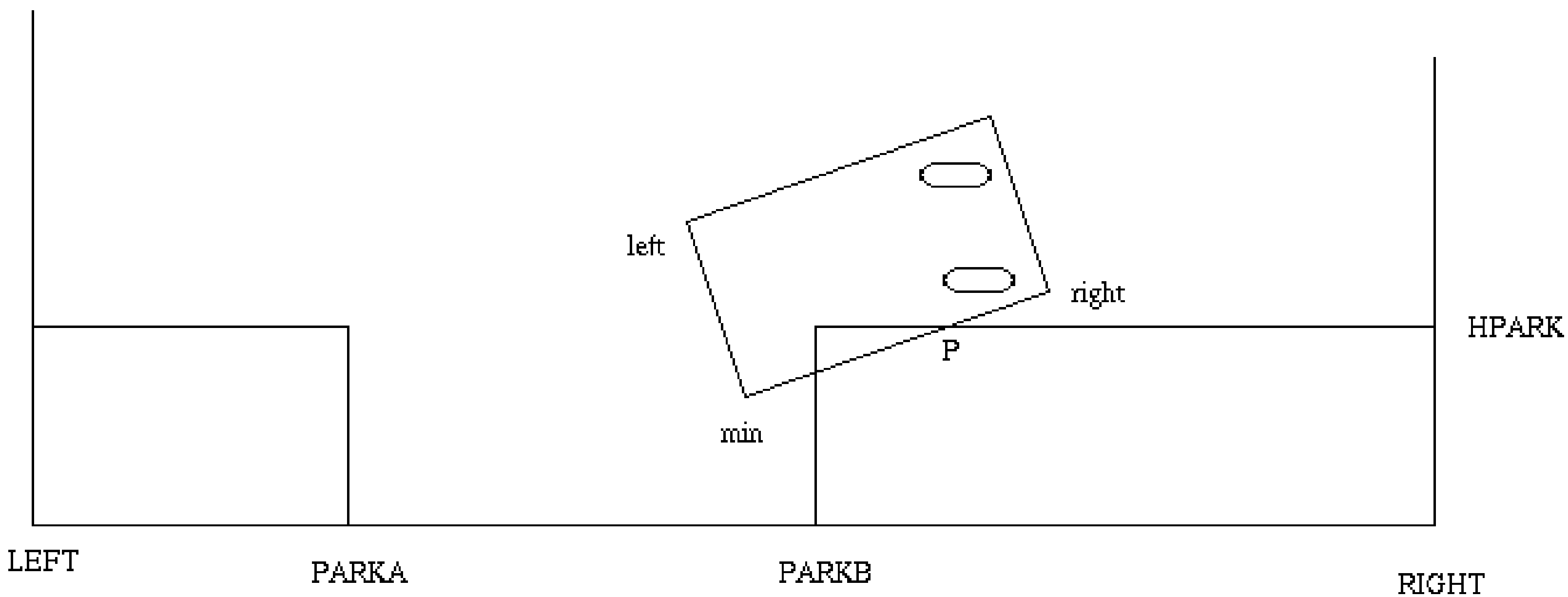
1	2	3
8		4
7	6	5

Ejemplo: Agente Estacionador de Autos

- Espacio de estados: (x,y, theta)
- Acciones: (v,psi)



Agente Estacionador de Autos: evitando colisiones



Agente Estacionador de Autos

- Basado en Técnicas de Búsqueda en Grafos: se uso una variante del algoritmo A*
- Estacionada convencional
- Estacionada no convencional
- Plan ineficiente

IA y Búsqueda en Grafos

- Robótica: planificación de acciones en robots.
- Juegos: puzzles, Backgammon, Ajedrez,
- Problemas combinatorios: programación, logística, problemas de optimización.
- Demostraciones matemáticas y lógicas.
  - Resolución es un problema de búsqueda.
- Diagnósticos en sistemas expertos
- MUCHOS problemas en IA pueden ser modelados como búsqueda en grafos

Grafos

- Grafo dirigido: (N,E):
  - N es un conjunto de nodos, representan estados;
  - E es un conjunto de arcos dirigidos entre nodos; representan acciones.
- Nodos y arcos están etiquetados con propiedades:
  - Propiedades de nodos: ej., valore, atributos, etc.
  - Propiedades de arcos: costo de la acción, atributos, etc.
- Árbol:
  - Nodo raíz, no está conectado desde ningún nodo
  - Cada nodo está directamente conectado desde un único nodo
- Camino: secuencia de nodos (n_1,…,n_k) donde n_i está directamente conectado a n_{i+1}
- Ciclo: camino (n_1,…, n_k) donde n_1=n_k
- Camino y Grafo acíclico: no tienen ciclos

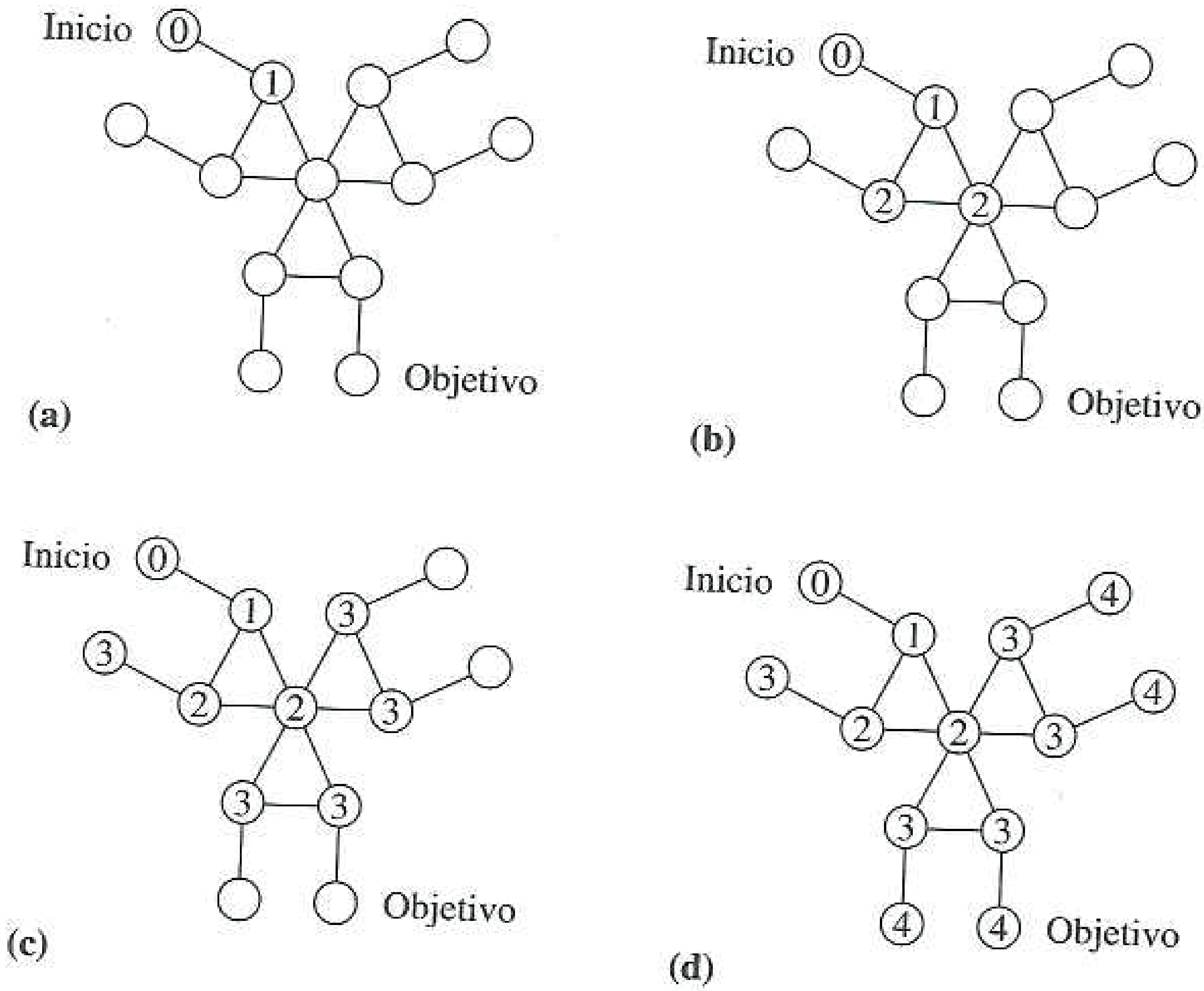
Problema de Búsqueda en Grafos

- Dado un grafo (N,E), un nodo de inicio s in N, un conjunto de nodos objetivos *G* subset N, encontrar un camino desde *s* a algún nodo en *G* que satisfaga una propiedad X:
- Propiedades posibles:
  - X = "null": cualquier camino sirve
  - X= "P es el mejor camino a un nodo en *G*": criterio de optimización
    - Ejemplo: camino más corto, menor costo, más rápido, etc.
  - X = "P tiene calidad mayor que q": problema de "satisfacción"
    - Ejemplo: camino que me lleve de un punto a otro en menos de q minutos.

Grafos Implícitos vs. Explícitos

- Los problemas de búsqueda más interesantes involucran grafos implícitos:
  - No siempre posible construir/almacenar grafos de estados,
    - Ej., Ajedrez tiene aprox. 10^10 estados.
  - Mejor generar estados vecinos a medida que recorremos el grafo
  - Requiere definir qué acciones usar para generar vecinos

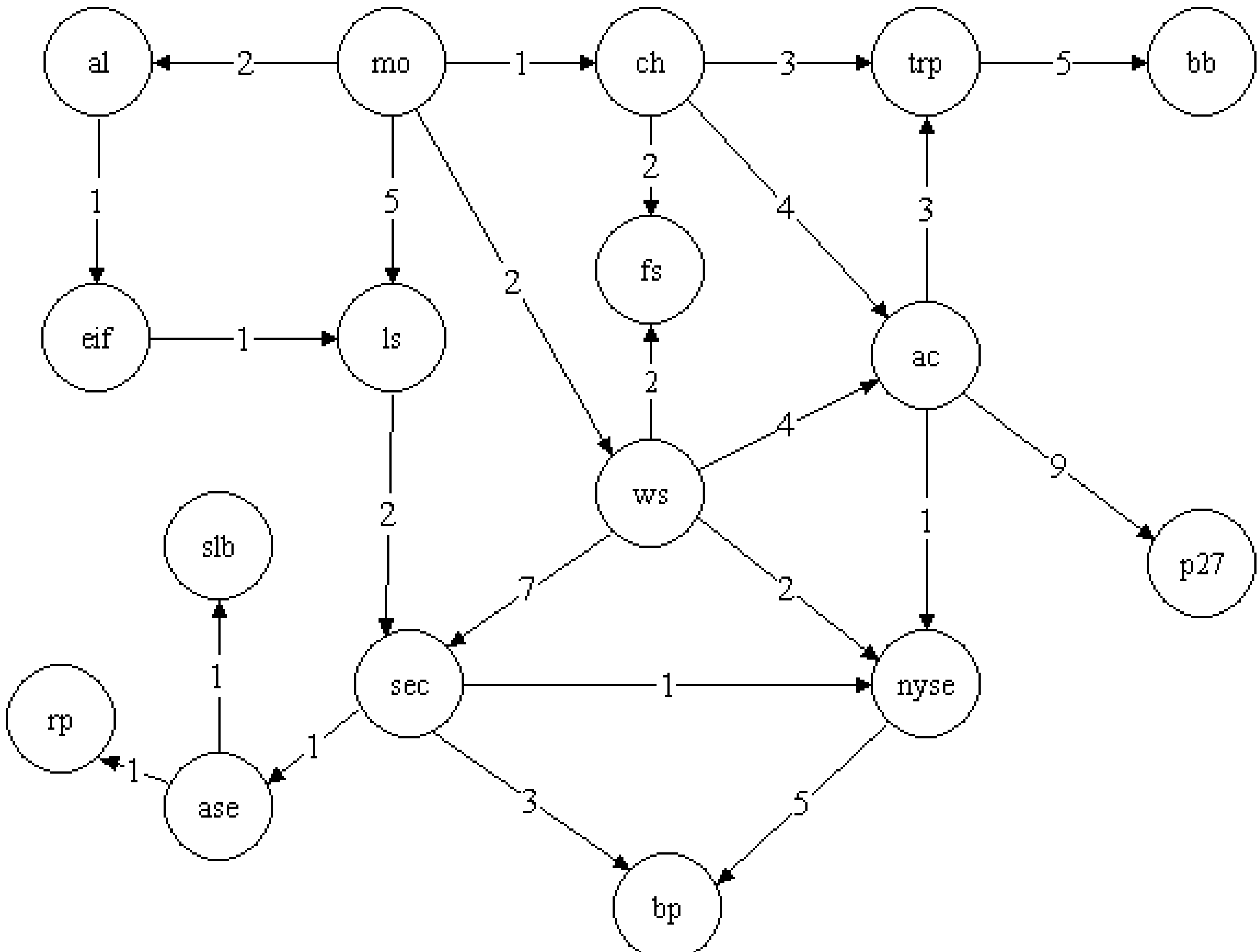
Algoritmo Genérico de Búsqueda



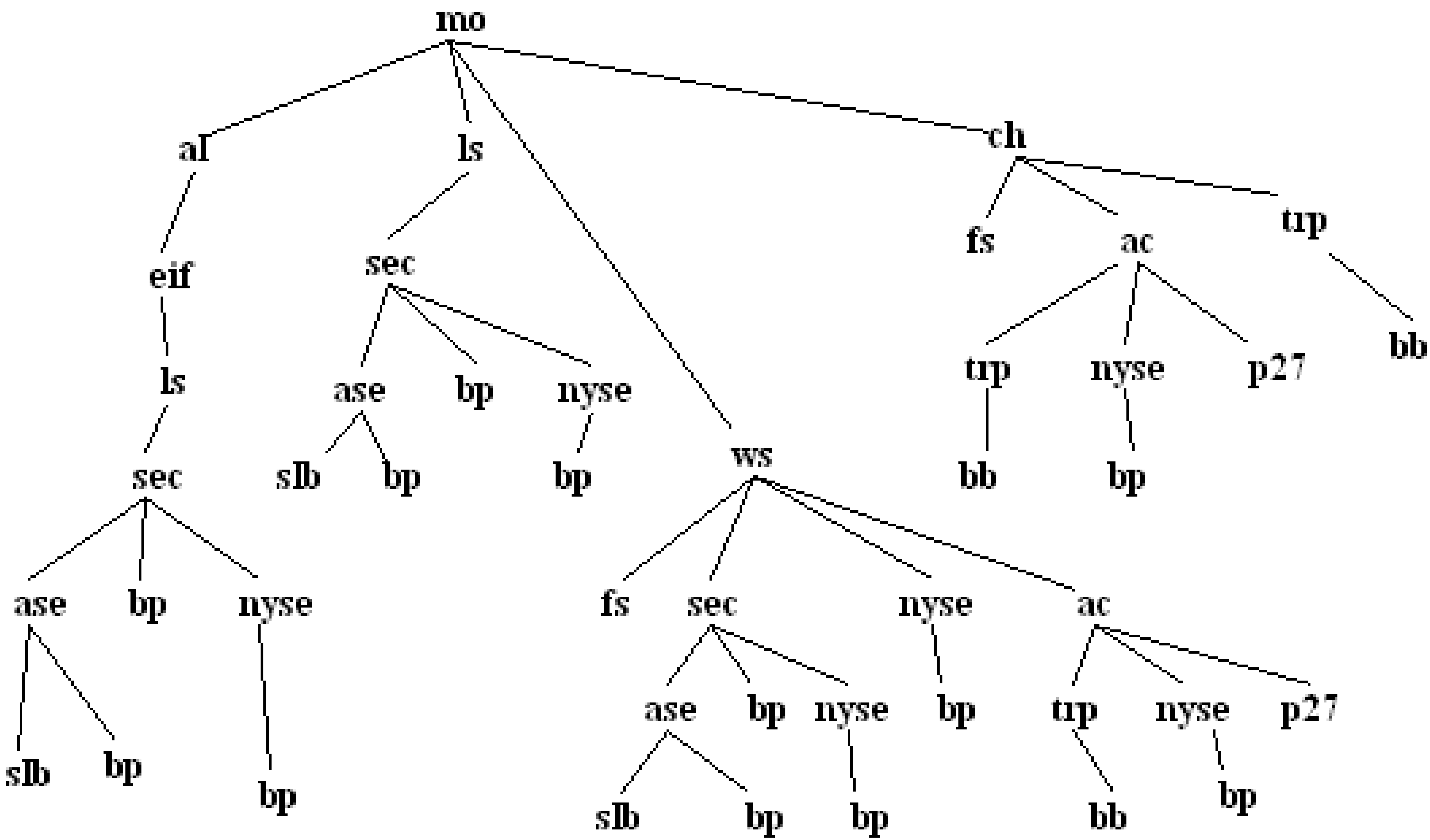
Algoritmo Genérico de Búsqueda

- Input: (V,E), s, G, X
- Variables:
  - FRONTERA: lista de nodos a visitar, inicialmente s
  - Tr: árbol de búsqueda, originalmente contiene un nodo etiquetado con s
- While (FRONTERA no es vacío) do
  - Sacar primer nodo n de FRONTERA
  - Si n está en G y su camino satisface X stop, entregar solución
  - Agregar al final de FRONTERA nodos P apuntados por n
  - Por cada nodo r en P, agregar un nuevo nodo a Tr, etiquetarlo con r y conectarlo desde el nodo de Tr asociado a n
  - Reordenar FRONTERA de acuerdo a algún esquema

Manhattan Bike Curier (Acíclico) Ref. Curso IA U. of Toronto



Arbol de Búsqueda (s = mo)



Árbol de búsqueda

- Guarda los caminos generados (recordar que estamos buscando caminos)
- Un nodo de grafo puede aparecer más de una vez en el árbol de búsqueda
- El árbol de búsqueda puede ser mucho más grande que el grafo original
- Al nivel k del árbol de búsqueda:
  - se tienen todos los estados que se pueden alcanzar desde s en k pasos
  - Todo camino de largo k desde s en el grafo es un camino en el árbol de búsqueda
- FRONTERA contiene las hojas del árbol

Estrategias de Búsqueda

- Búsqueda Primero en Anchura
- Búsqueda Primero en Profundidad

Búsqueda Primero en Anchura (breadth-first search)

- El árbol de búsqueda se genera a lo ancho
- Variante: búsqueda de costo uniforme (Dijkstra, 1959)
  - Se expanden los nodos de igual costo
- Costo en memoria:
  - El árbol es exponencial en su profundidad máxima

Ejemplo: puzzle de ocho piezas

Estado Inicial

2	8	3
1	6	4
7		5

Objetivo

1	2	3
8		4
7	6	5

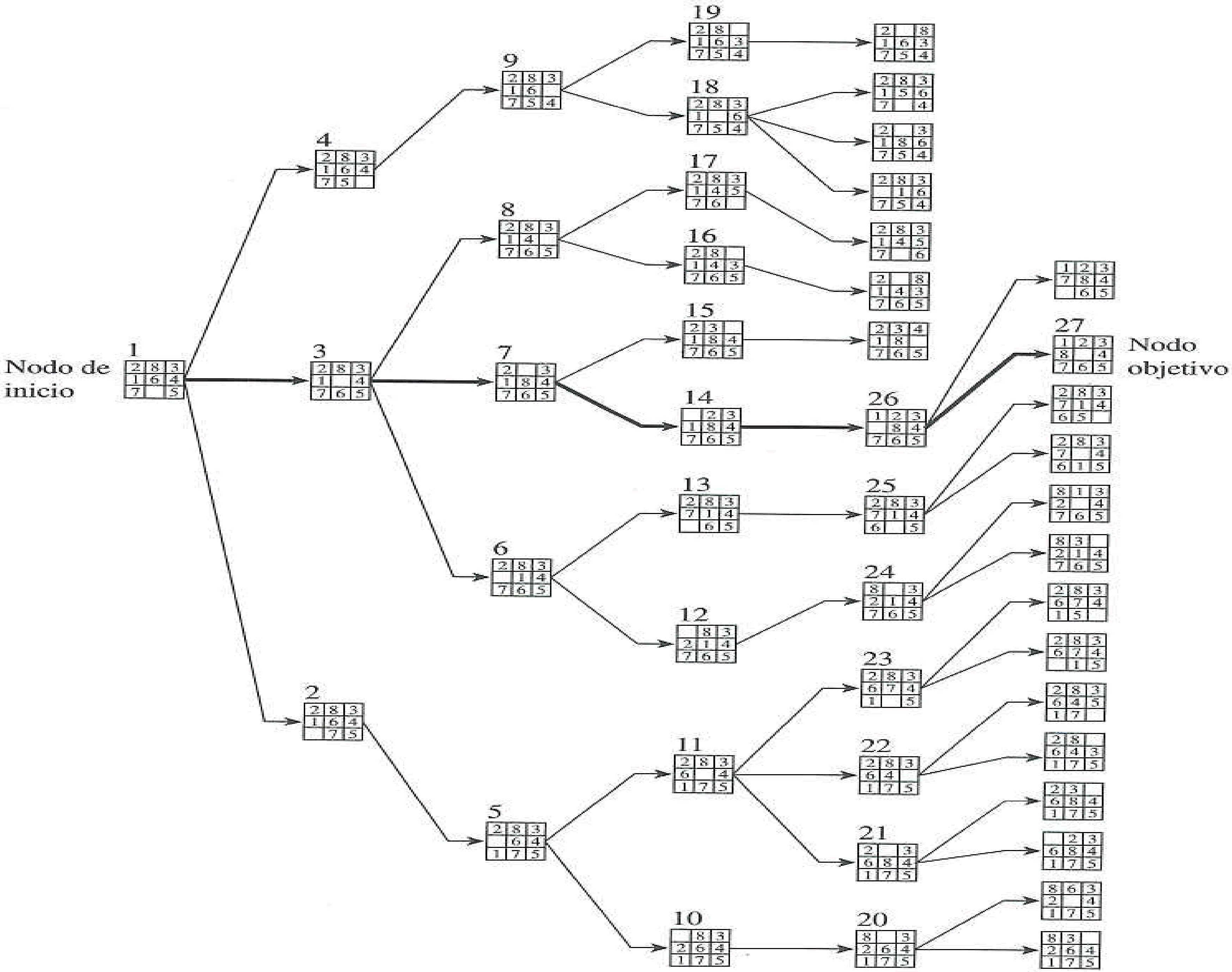
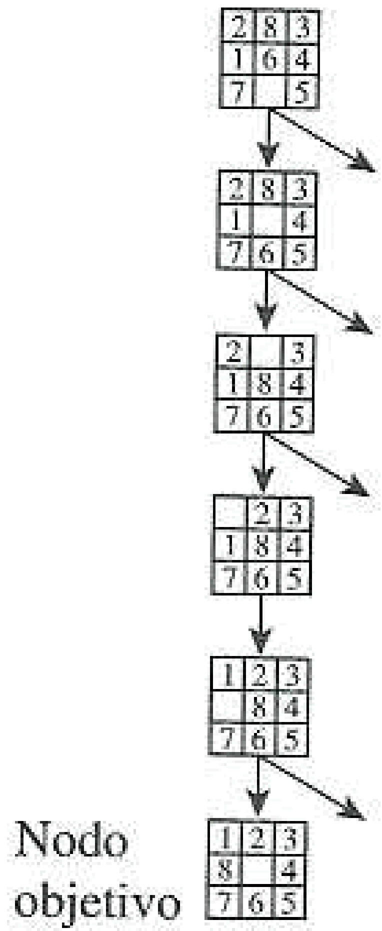

Nodo de inicio
Nodo objetivo
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

Figura 8.2.

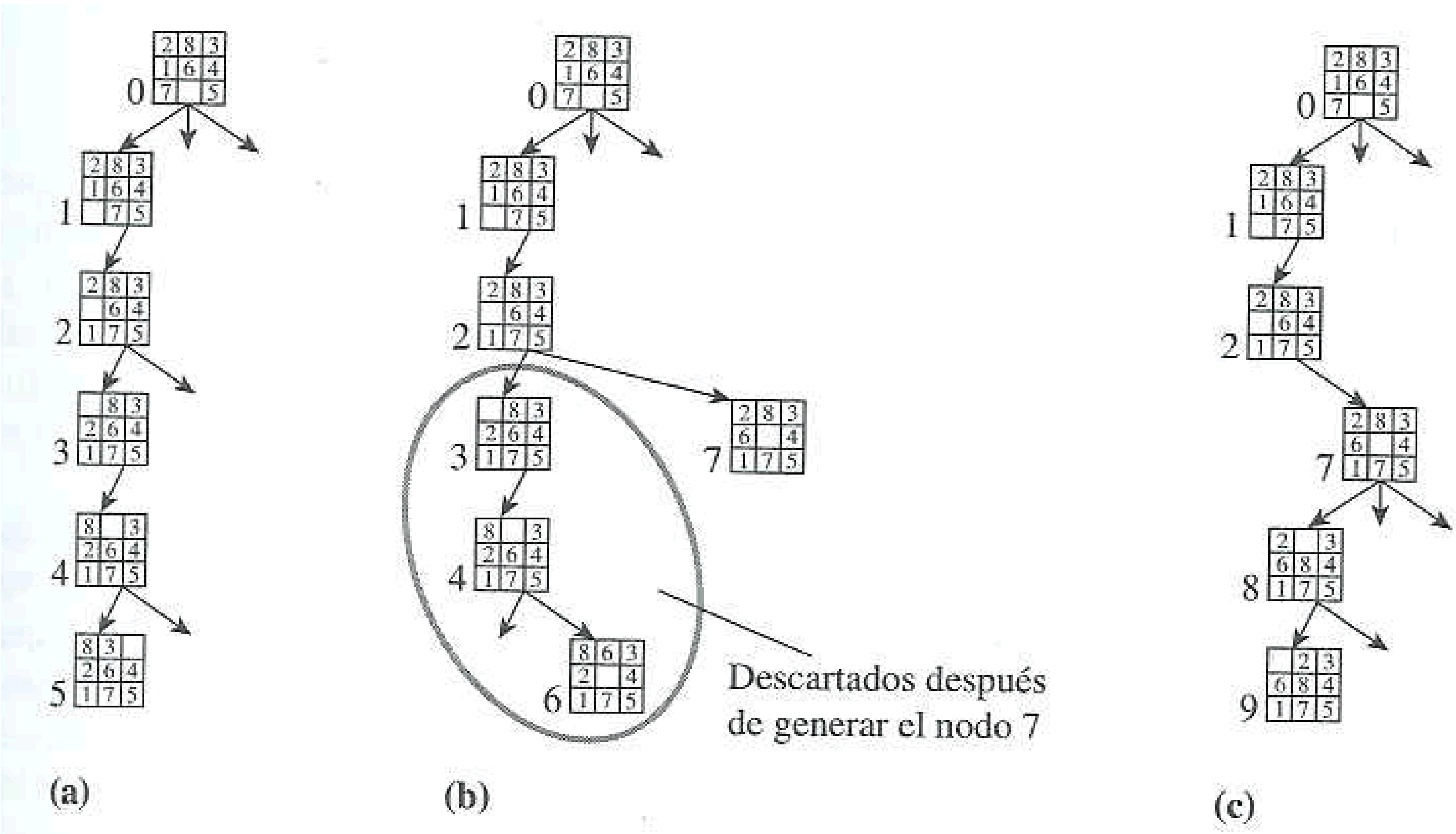
Búsqueda primero en profundidad (depth-first search)

- El árbol de búsqueda se genera en profundidad y sólo almacenamos un solo camino en el.
- Usualmente se fija una profundidad límite
- El tamaño del árbol de búsqueda es a lo más lineal en la profundidad límite.
- Requiere saltos hacia atrás (backtracking)
  - Ejemplo, si se llega a una profundidad límite saltar hacia atrás.
- Sin embargo, si encontramos el nodo objetivo, no podemos asegurar que es el de menor profundidad (camino más corto).

Ejemplo: búsqueda en profundidad



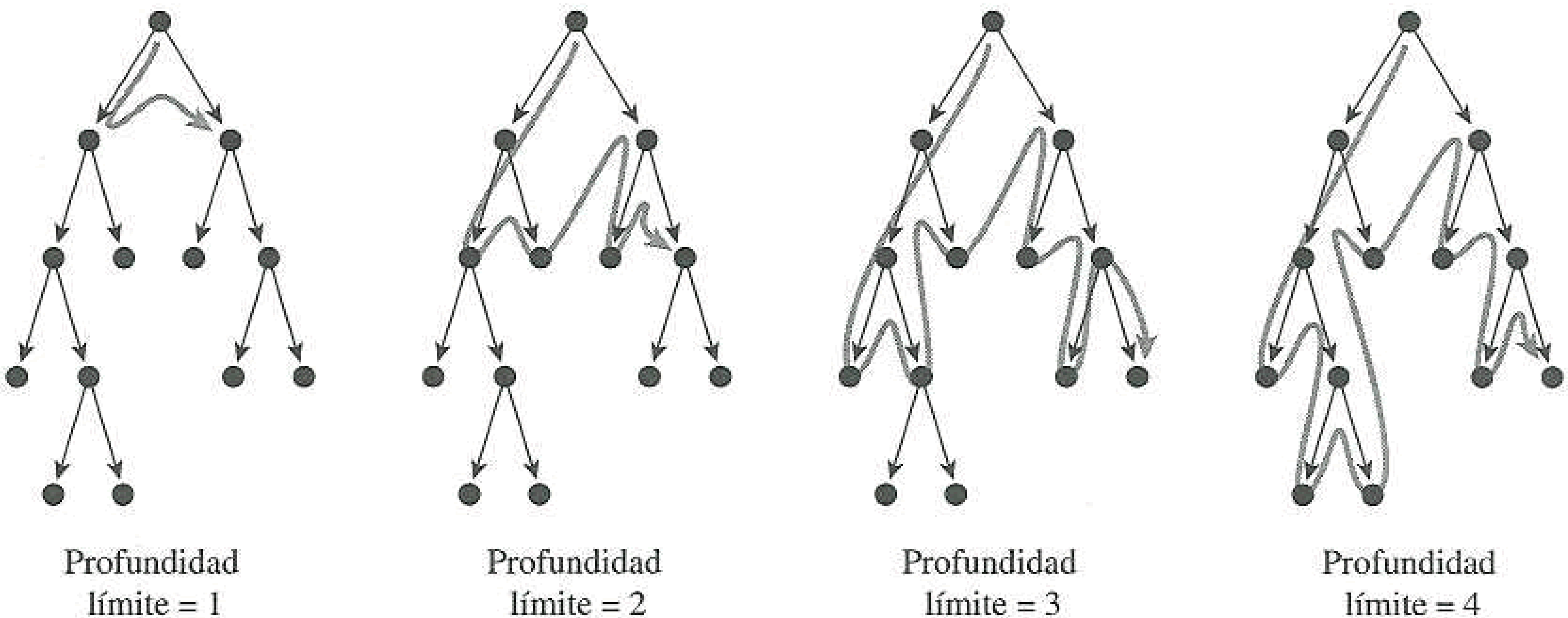
Ejemplo: búsqueda en profundidad con profundidad límite 5



Descenso Iterativo

- En cada paso buscamos en profundidad con profundidad límite
- Después de cada paso aumentamos la profundidad límite
- Más adelante mostraremos que el número de nodos expandidos no es necesariamente mucho mayor que en la búsqueda en anchura

Descenso Iterativo (ejemplo)

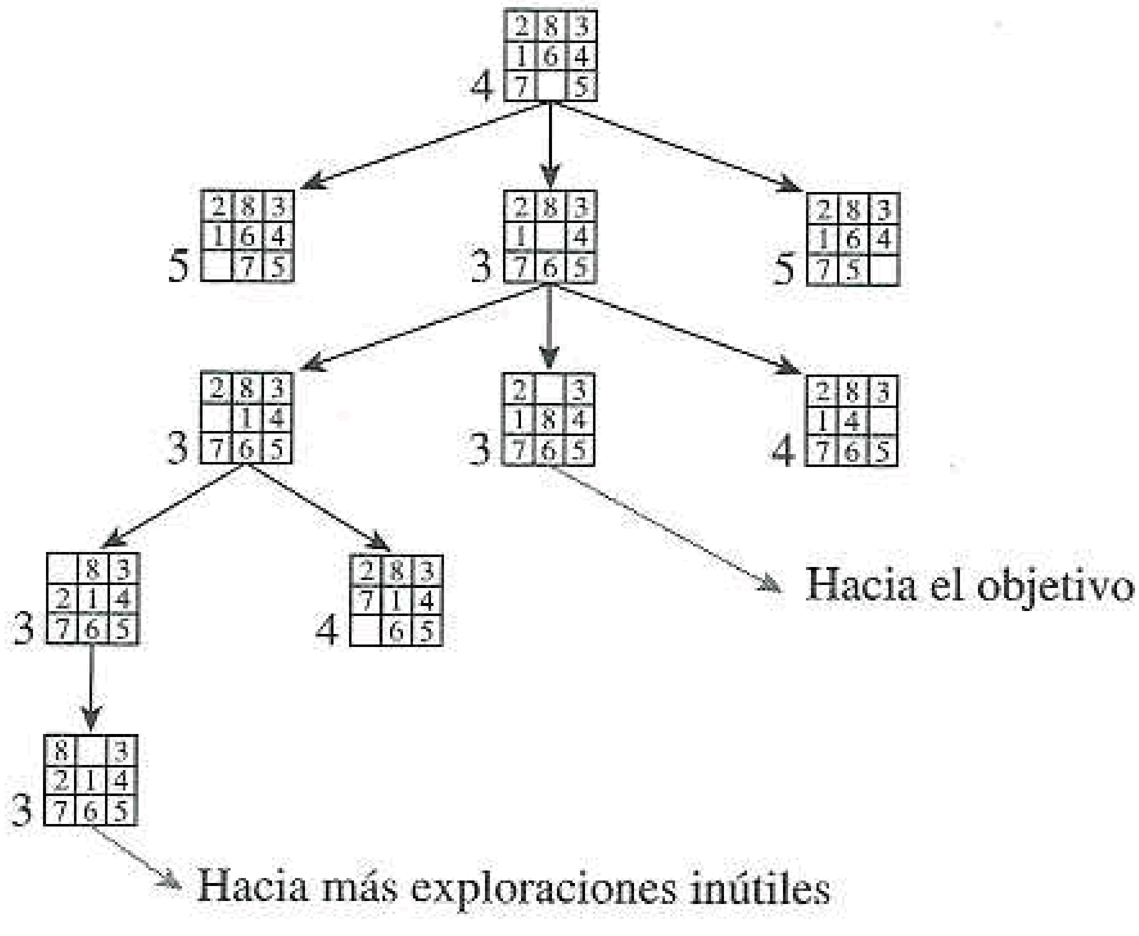


Búsqueda Heurística

- Tenemos una función f (función de evaluación) que determina cómo ordenamos la lista FRONTERA
- f es una función definida sobre las descripciones de los estados
- f define cuál es el mejor nodo para expandir
  - A menor f(n), expandir n es mejor
- En la mayoría de los casos es posible encontrar buenas funciones de evaluación

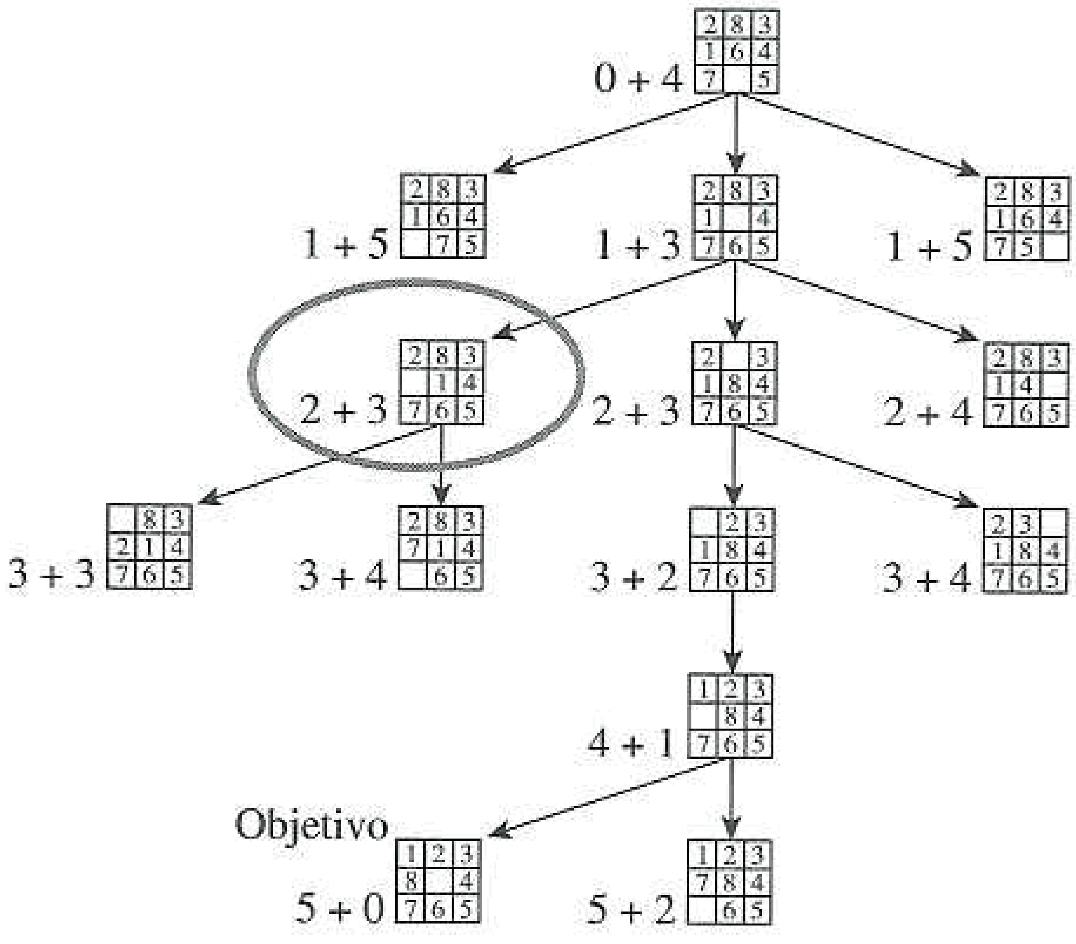
Búsqueda Heurística: Ejemplo

- f(n)= número de fichas descolocadas (comparadas con el estado objet



Factor de profundidad

- Funciona mejor $f(n) = g(n) + h(n)$



Algoritmo A*

- [Hart, Nilsson and Raphael, 1968]
- *Idea:*
  - *Algoritmo genérico, pero...*
  - *Reordenar FRONTERA de acuerdo a función f*

Algoritmo A*

- Input: (V,E), s, G, X
- Variables:
  - FRONTERA: lista de nodos a visitar, inicialmente s
  - Tr: árbol de búsqueda, originalmente contiene un nodo etiquetado con s
- While (FRONTERA no es vacío) do
  - Sacar primer nodo n de FRONTERA
  - Si n está en G (y su camino satisface X) stop, entregar solución
  - Agregar al final de FRONTERA nodos P apuntados por n
  - Por cada nodo r en P, agregar un nuevo nodo a Tr, etiquetarlo con r y conectarlo desde el nodo de Tr asociado a n
  - *Reordenar FRONTERA de acuerdo a función f*