

Inteligencia artificial

Una nueva síntesis



Nils J. Nilsson

INTELIGENCIA ARTIFICIAL

Una nueva síntesis

CONSULTORES EDITORIALES
ÁREA DE INFORMÁTICA Y COMPUTACIÓN

Antonio Vaquero Sánchez

Catedrático de Lenguaje y Sistemas Informáticos
Escuela Superior de Informática
Universidad Complutense de Madrid
ESPAÑA

Gerardo Quiroz Vieyra

Ingeniero de Comunicaciones y Electrónica
por la ESIME del Instituto Politécnico Nacional
Profesor de la Universidad Autónoma Metropolitana
Unidad Xochimilco
MÉXICO

INTELIGENCIA ARTIFICIAL

Una nueva síntesis

NILS J. NILSSON

Stanford University

Traducción

ROQUE MARÍN MORALES
JOSÉ TOMÁS PALMA MÉNDEZ
ENRIQUE PANIAGUA ARIS

Departamento de Informática, Inteligencia Artificial y Electrónica
Universidad de Murcia

Revisión técnica

SEBASTIÁN DORMIDO BENCOMO
Departamento de Informática y Automática
Universidad Nacional de Educación a Distancia

LIBRO DE CORTESIA



MEXICO

MADRID • BUENOS AIRES • CARACAS • GUATEMALA • LISBOA • MÉXICO
NUEVA YORK • PANAMÁ • SAN JUAN • SANTAFÉ DE BOGOTÁ • SANTIAGO • SÃO PAULO
AUCKLAND • HAMBURGO • LONDRES • MILÁN • MONTREAL • NUEVA DELHI • PARÍS
SAN FRANCISCO • SIDNEY • SINGAPUR • ST. LOUIS • TOKIO • TORONTO

INTELIGENCIA ARTIFICIAL. Una nueva síntesis

No está permitida la reproducción total o parcial de este libro, ni su tratamiento informático, ni la transmisión de ninguna forma o por cualquier medio, ya sea electrónico, mecánico, por fotocopia, por registro u otros métodos, sin el permiso previo y por escrito de los titulares del Copyright.

DERECHOS RESERVADOS © 2001, respecto a la primera edición en español, por
McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U.
Edificio Valrealty, 1.^a planta
Basauri, 17
28023 Aravaca (Madrid)

Traducido de la primera edición en inglés de:
ARTIFICIAL INTELLIGENCE: A NEW SYNTHESIS

Copyright © MCMXCVIII by Morgan Kaufmann Publishers, Inc.
ISBN: 1-55860-467-7

ISBN: 84-481-2824-9
Depósito legal: M. 33.274-2000

Editora: Concepción Fernández Madrid
Cubierta: Design Master. Dima
Compuesto en C + I, S. L.
Impreso en Impresos y Revistas, S. A. (IMPRESA)

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

A Scott y Ryan

Contenido

Prefacio	xvii
1. Introducción	1
1.1. ¿Qué es la IA?	1
1.2. Aproximaciones a la Inteligencia Artificial	5
1.3. Breve historia de la IA	7
1.4. Plan del libro	10
1.5. Lecturas y consideraciones adicionales	12
Ejercicios	15
I. SISTEMAS REACTIVOS	17
2. Agentes de estímulo-respuesta	19
2.1. Percepción y acción	19
2.1.1. Percepción	22
2.1.2. Acción	22
2.1.3. Álgebra booleana	23
2.1.4. Clases y formas de las funciones booleanas	24
2.2. Representación e implementación de las funciones para la selección de acciones	24
2.2.1. Sistemas de producción	25
2.2.2. Redes	26
2.2.3. La arquitectura de subsunción	29
2.3. Lecturas y consideraciones adicionales	30
Ejercicios	31

3. Redes neuronales	33
3.1. Introducción	33
3.2. Entrenamiento de una ULU	34
3.2.1. Interpretación geométrica del funcionamiento de una ULU	34
3.2.2. La dimensión $n + 1$	35
3.2.3. Métodos del gradiente descendente	35
3.2.4. El procedimiento de Widrow-Hoff	37
3.2.5. El procedimiento delta generalizado	37
3.2.6. El procedimiento de corrección del error	39
3.3. Redes neuronales	40
3.3.1. Motivación	40
3.3.2. Notación	41
3.3.3. El método de la retropropagación	42
3.3.4. Cálculo del cambio de los pesos en la última capa	43
3.3.5. Cálculo del cambio de los pesos en las capas intermedias	44
3.4. Generalización, precisión y sobreajuste	46
3.5. Lecturas y consideraciones adicionales	50
Ejercicios	51
4. Sistemas evolutivos	53
4.1. Computación evolutiva	53
4.2. Programación genética	54
4.2.1. Representación de los programas en PG	54
4.2.2. El proceso PG	57
4.2.3. Desarrollo de un robot seguidor de paredes	58
4.3. Lecturas y consideraciones adicionales	62
Ejercicios	62
5. Sistemas con estados	63
5.1. Representación del entorno mediante vectores de características	63
5.2. Redes de Elman	65
5.3. Representaciones icónicas	66
5.4. Sistemas basados en pizarras	69
5.5. Lecturas y consideraciones adicionales	71
Ejercicios	71
6. Visión artificial	75
6.1. Introducción	75
6.2. Conduciendo un automóvil	76
6.3. Las dos etapas de la visión por computador	78
6.4. Procesamiento de imágenes	80
6.4.1. Promediado	80
6.4.2. Detección de bordes	83

6.4.3. Un operador para el promediado y la detección de bordes	85
6.4.4. Búsqueda de regiones	86
6.4.5. Otros atributos de imagen distintos a la intensidad	89
6.5. Análisis de escenas	91
6.5.1. Interpretación de líneas y curvas en una imagen	92
6.5.2. Visión basada en modelos	94
6.6. Visión estereoscópica e información sobre la intensidad	96
6.7. Lecturas y consideraciones adicionales	98
Ejercicios	99
II. BÚSQUEDA EN ESPACIOS DE ESTADO	103
7. Agentes que planifican	105
7.1. Memoria o cálculo	105
7.2. Grafos de estados	106
7.3. Búsqueda en estados explícitos	109
7.4. Estados basados en características	110
7.5. Notación de grafos	111
7.6. Lecturas y consideraciones adicionales	112
Ejercicios	112
8. Búsqueda a ciegas	115
8.1. Especificación del espacio de búsqueda	115
8.2. Elemento de un grafo de estados implícito	116
8.3. Búsqueda primero en anchura	117
8.4. Búsqueda primero en profundidad o búsqueda con vuelta atrás	118
8.5. Descenso iterativo	120
8.6. Lecturas y consideraciones adicionales	122
Ejercicios	122
9. Búsqueda heurística	125
9.1. Funciones de evaluación	125
9.2. Algoritmo genérico de búsqueda en grafos	127
9.2.1. Algoritmo A*	128
9.2.2. Admisibilidad del algoritmo A*	131
9.2.3. La condición de consistencia o monotomía	134
9.2.4. Descenso iterativo A*	138
9.2.5. Búsqueda «primero el mejor» recursiva	139
9.3. Funciones heurísticas y eficiencia del proceso de búsqueda	140
9.4. Lecturas y consideraciones adicionales	143
Ejercicios	144

10. Planificación, actuación y aprendizaje	147
10.1. El ciclo percibir/planificar/actuar	147
10.2. Búsqueda aproximada	149
10.2.1. Búsqueda orientada a subobjetivos	149
10.2.2. Búsqueda jerárquica	150
10.2.3. Búsqueda con horizonte	152
10.2.4. Ciclos	153
10.2.5. Construcción de procedimientos reactivos	153
10.3. Aprendizaje de funciones heurísticas	155
10.3.1. Grafos explícitos	155
10.3.2. Grafos implícitos	156
10.4. Recompensas en vez de objetivos	158
10.5. Lecturas y consideraciones adicionales	160
Ejercicios	160
11. Métodos alternativos de búsqueda y otras aplicaciones	163
11.1. Problemas de asignación	163
11.2. Métodos constructivos	165
11.3. Reparación heurística	169
11.4. Optimización de funciones	171
Ejercicios	174
12. Búsqueda en problemas de juegos	175
12.1. Juegos de dos jugadores	175
12.2. El procedimiento minimax	177
12.3. El procedimiento alfa-beta	181
12.4. Eficiencia del procedimiento alfa-beta	186
12.5. Otras cuestiones importantes	187
12.6. Juegos de azar	187
12.7. Aprendizaje de funciones de evaluación	189
12.8. Lecturas y consideraciones adicionales	190
Ejercicios	191
III. REPRESENTACIÓN DEL CONOCIMIENTO Y RAZONAMIENTO	193
13. El cálculo proporcional	195
13.1. Imponiendo restricciones a los valores de las características	195
13.2. El lenguaje	197
13.3. Las reglas de inferencia	198
13.4. Definición de demostración	198

13.5. La semántica	199
13.5.1. Interpretaciones	199
13.5.2. La tabla de verdad proposicional	200
13.5.3. Satisfacibilidad y modelos	201
13.5.4. Validez	202
13.5.5. Equivalencia	202
13.5.6. Consecuencia lógica	202
13.6. Solidez y completitud	203
13.7. El problema de SATP	204
13.8. Otras cuestiones importantes	205
13.8.1. Distinción entre los lenguajes	205
13.8.2. Metateoremas	205
13.8.3. Leyes asociativas	205
13.8.4. Leyes distributivas	206
Ejercicios	206
14. La resolución en el cálculo proposicional	207
14.1. Una nueva regla de inferencia: la resolución	207
14.1.1. Las cláusulas como fbsf	207
14.1.2. La resolución aplicada a las cláusulas	207
14.1.3. La corrección de la resolución	208
14.2. Conversión de fbsf a conjunciones de cláusulas	208
14.3. Refutación mediante resolución	209
14.4. Estrategias de búsqueda en refutación mediante resolución	211
14.4.1. Estrategias de ordenación	211
14.4.2. Estrategias de refinamiento	212
14.5. Cláusulas de Horn	212
Ejercicios	213
15. El cálculo de predicados	215
15.1. Introducción	215
15.2. El lenguaje y su sintaxis	216
15.3. La semántica	217
15.3.1. Mundos	217
15.3.2. Interpretaciones	218
15.3.3. Modelos y otros conceptos relacionados	219
15.3.4. Conocimiento	219
15.4. Cuantificación	220
15.5. Semántica de los cuantificadores	222
15.5.1. Cuantificadores universales	222
15.5.2. Cuantificadores existenciales	222
15.5.3. Equivalencias útiles	222
15.5.4. Reglas de inferencia	222

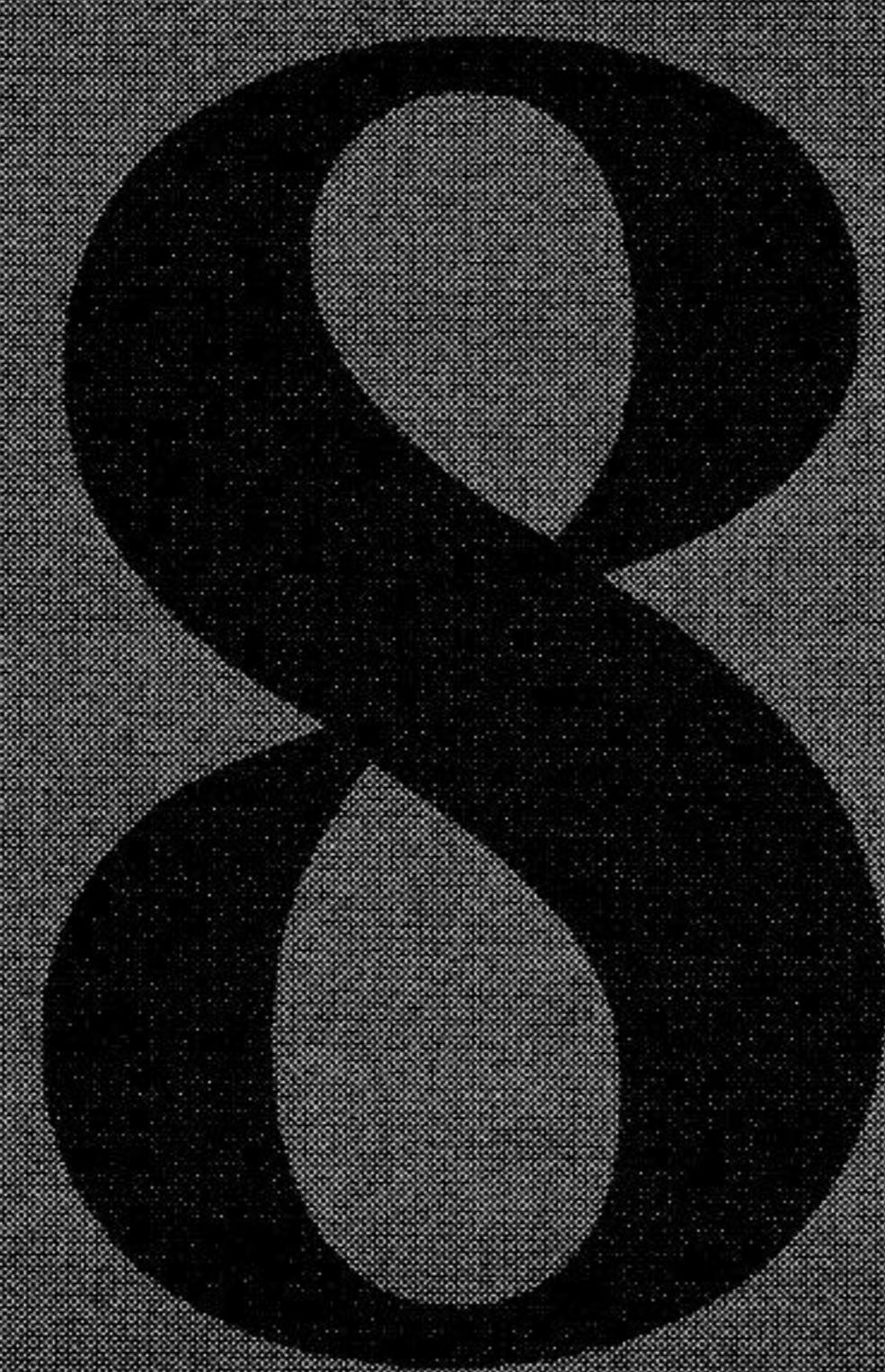
15.6. El cálculo de predicados como un lenguaje de representación del conocimiento.	223
15.6.1. Conceptualizaciones	223
15.6.2. Ejemplos	223
15.7. Lecturas y consideraciones adicionales	225
Ejercicios	225
16. La resolución en el cálculo de predicados	227
16.1. Unificación	227
16.2. Resolución en el cálculo de predicados	230
16.3. Completitud y solidez	230
16.4. Conversión de fbfs arbitrarias en cláusulas	231
16.5. Utilización de la resolución para demostrar teoremas	233
16.6. Obtención de respuestas	234
16.7. El predicado de igualdad	235
16.8. Lecturas y consideraciones adicionales	237
Ejercicios	238
17. Sistemas basados en conocimiento	241
17.1. El enfrentamiento con el mundo	241
17.2. Razonamiento con cláusulas de Horn	242
17.3. Mantenimiento de bases de conocimiento dinámicas	247
17.4. Sistemas expertos basados en reglas	251
17.5. Aprendizaje de reglas	255
17.5.1. Aprendizaje de reglas de cálculo proposicional	256
17.5.2. Aprendizaje de reglas en lógica de predicados	260
17.5.3. Generalización basada en explicaciones	264
17.6. Lecturas y consideraciones adicionales	265
Ejercicios	266
18. Representación del sentido común	269
18.1. El mundo del sentido común	269
18.1.1. ¿Qué es el conocimiento del sentido común?	269
18.1.2. Problemas de la representación del conocimiento del sentido común	271
18.1.3. La importancia del conocimiento del sentido común	272
18.1.4. Áreas de investigación	272
18.2. El tiempo	273
18.3. Representación del conocimiento mediante redes	275
18.3.1. Conocimiento taxonómico	275
18.3.2. Las redes semánticas	276
18.3.3. Razonamiento no monótono en redes semánticas	277
18.3.4. Guiones	279
18.4. Lecturas y consideraciones adicionales	280
Ejercicios	281

19. Razonamiento con incertidumbre	285
19.1. Repaso a la teoría de probabilidades	285
19.1.1. Conceptos fundamentales	285
19.1.2. Probabilidades condicionales	288
19.2. Inferencia probabilística	290
19.2.1. Un método general	290
19.2.2. Independencia condicional	292
19.3. Redes bayesianas	293
19.4. Patrones de inferencia en redes bayesianas	295
19.5. Evidencia con incertidumbre	296
19.6. Separación- <i>d</i>	297
19.7. Inferencia probabilística en poliárboles	298
19.7.1. Apoyo causal	299
19.7.2. Apoyo evidencial	300
19.7.3. Apoyos causal y evidencial	302
19.7.4. Un ejemplo numérico	302
19.8. Lecturas y consideraciones adicionales	304
Ejercicios	305
20. Aprendizaje y actuación con redes bayesianas	309
20.1. Aprendizaje de redes bayesianas	309
20.1.1. Una estructura de red conocida	309
20.1.2. Aprendizaje de la estructura de la red	312
20.2. Inferencia probabilística y actuación	316
20.2.1. El escenario genérico	316
20.2.2. Un ejemplo extendido	317
20.2.3. Generalización del ejemplo	321
20.3. Lecturas y consideraciones adicionales	322
Ejercicios	322
IV. MÉTODOS DE PLANIFICACIÓN BASADOS EN LÓGICA	325
21. El cálculo de situaciones	327
21.1. Razonamiento acerca de estados y situaciones	327
21.2. Algunas dificultades	330
21.2.1. Axiomas de marco	330
21.2.2. Cualificaciones	332
21.2.3. Ramificaciones	332
21.3. Generación de planes	333
21.4. Lecturas y consideraciones adicionales	333
Ejercicios	334

22. Planificación	337
22.1. Sistemas de planificación basados en STRIPS	337
22.1.1. Descripción de estados y objetivos	337
22.1.2. Métodos de búsqueda hacia delante	338
22.1.3. STRIPS recursivo	341
22.1.4. Planes con condiciones en tiempo de ejecución	343
22.1.5. La anomalía de Sussman	344
22.1.6. Métodos de búsqueda hacia atrás	344
22.2. Espacio de planes y planificación parcialmente ordenada	348
22.3. Planificación jerárquica	354
22.3.1. ABSTRIPS	354
22.3.2. Combinación de la planificación jerárquica y la planificación parcialmente ordenada	356
22.4. Aprendizaje de planes	357
22.5. Lecturas y consideraciones adicionales	359
Ejercicios	361
V. COMUNICACIÓN E INTEGRACIÓN	365
23. Múltiples agentes	367
23.1. Interacción entre agentes	367
23.2. Modelos de otros agentes	368
23.2.1. Tipos de modelos	368
23.2.2. Estrategias de simulación	369
23.2.3. Bases de datos simuladas	369
23.2.4. La actitud intencional	370
23.3. Una lógica modal para representar el conocimiento	371
23.3.1. Operadores modales	371
23.3.2. Axiomas del conocimiento	372
23.3.3. Razonamiento acerca del conocimiento de otros agentes	374
23.3.4. Predicción de las acciones de otros agentes	375
23.4. Lecturas y consideraciones adicionales	376
Ejercicios	376
24. Comunicación entre agentes	379
24.1. Actos de habla	379
24.1.1. Planificación de los actos de habla	380
24.1.2. Implementación de los actos de habla	381
24.2. Comprensión de las cadenas de símbolos de un lenguaje	383
24.2.1. Gramáticas con estructura de frase	383
24.2.2. Análisis semántico	386
24.2.3. Extensión de la gramática	390
24.3. Comunicación eficiente	392
24.3.1. Utilización del contexto	392
24.3.2. Utilización de conocimiento para resolver ambigüedades	393

24.4. Procesamiento del lenguaje natural	394
24.5. Lecturas de consideraciones adicionales	397
Ejercicios	397
25. Arquitecturas de agente	399
25.1. Arquitectura de tres niveles	400
25.2. Evaluación y selección de objetivos	402
25.3. Arquitectura de tres módulos	403
25.4. Introspección y evolución	404
25.5. Lecturas y consideraciones adicionales	405
Ejercicios	406
Bibliografía	407
Índice	433

Búsqueda a ciegas



8.1. *Especificación del espacio de búsqueda*

Muchos de los problemas de interés práctico tienen unos espacios de búsqueda tan grandes que no pueden ser representados mediante un grafo explícito, por tanto, se requieren modificaciones en los procedimientos de búsqueda tratados en el capítulo anterior. Primero, debemos prestar una atención especial a la formulación de dichos problemas para poder aplicar sobre ellos los métodos de búsqueda. Segundo, necesitamos métodos que nos permitan representar grandes espacios de búsqueda mediante grafos implícitos. Tercero, necesitamos métodos eficientes de búsqueda en esos grafos de gran tamaño. En algunos problemas de planificación, como, por ejemplo, el del apilamiento de bloques, no es difícil concebir estructuras de datos para representar los diferentes estados del mundo y las acciones que lo modifican. Normalmente, resulta difícil encontrar representaciones en forma de grafos de estados que sean manejables. Para conseguir esto, se requiere un análisis detallado del problema —buscar simetrías, ignorar detalles irrelevantes y encontrar las abstracciones apropiadas—. Desgraciadamente, plantear un problema para poder aplicar sobre él técnicas de búsqueda es una tarea bastante compleja y se puede considerar como un arte que todavía requiere de intervención humana.

Además del problema del apilamiento de bloques, también se utiliza a menudo el problema del desplazamiento de las piezas de un puzzle para mostrar cómo se pueden utilizar las técnicas de búsqueda en espacios de estados para planificar secuencias de acciones. Para ilustrar las técnicas de búsqueda, que examinaremos en este capítulo y en el siguiente, utilizaremos este tipo de problemas. Un ejemplo típico es el puzzle de 15 piezas, que consiste en un conjunto de 15 piezas dispuestas en un tablero de cuatro por cuatro, en el que se deja una casilla en blanco a la que se pueden desplazar las piezas colocadas en casillas adyacentes. El objetivo es encontrar una secuencia de movimientos que transforme una disposición inicial de piezas en una configuración determinada. El puzzle de ocho piezas es una versión reducida del problema, en el que sólo se tienen ocho piezas en un tablero de tres por tres. Supongamos que el objetivo del puzzle es mover las casillas desde una disposición inicial arbitraria hasta la configuración objetivo, tal y como se muestra en la Figura 8.1.

2	8	3
1	6	4
7		5

→

1	2	2
8		4
7	6	5

Figura 8.1.

Configuración inicial y objetivo para el problema del puzzle de ocho piezas.

Dado este problema, una descripción obvia que permite representar los distintos estados es una matriz de tres por tres, en la que cada celda contiene un número del 1 al 8 o un símbolo que represente la celda vacía. El estado objetivo es la matriz representada en la parte derecha de la Figura 8.1. Para pasar de un estado a otro sólo tenemos que desplazar una pieza a la celda vacía. Sin embargo, como ya hemos indicado, podemos elegir entre diferentes formas de representar el espacio de estados. En el problema del puzzle de ocho piezas, podríamos haber considerado que tenemos 8×4 movimientos posibles: desplazarse una casilla hacia arriba, una hacia abajo, una hacia la izquierda, una hacia la derecha, dos casillas hacia arriba... (por supuesto, no todos los movimientos son posibles en todos los estados). Una formulación más compacta sólo tiene en cuenta cuatro posibles movimientos: mover la casilla libre hacia la izquierda, hacia arriba, hacia la derecha y hacia abajo. El grafo con los estados accesibles desde el estado inicial queda definido de forma implícita a partir del estado inicial y del conjunto de movimientos posibles. El conjunto de nodos en el grafo de estados para esta representación del puzzle de ocho piezas es $9! = 362.880$ (en este problema, el grafo de estados puede ser dividido en dos grafos independientes, tales que una configuración de piezas perteneciente a uno de los grafos no puede ser alcanzada desde ninguno de los nodos del otro grafo).

8.2.

Elementos de un grafo de estados implícito

Como hemos visto en el párrafo anterior, describiendo el estado inicial y los efectos que las acciones producen sobre dicho estado, obtenemos una representación implícita de la parte del grafo de estados que se puede alcanzar a partir del estado inicial. Por tanto, y en principio, es posible transformar la representación implícita del grafo en un grafo explícito. Para hacerlo, sólo hay que generar todos los nodos sucesores del nodo inicial (aplicando todos los posibles operadores al nodo inicial) y, después, volver a realizar la misma operación para cada uno de los nodos sucesores. Para aquellos grafos que sean demasiado grandes para ser representados de forma explícita, el proceso de búsqueda sólo requiere tener de forma explícita la parte del espacio de estados necesaria para calcular el camino hacia el objetivo. El proceso termina cuando se ha encontrado un camino aceptable al nodo objetivo.

Formalmente, podemos decir que existen tres elementos básicos en la representación implícita del grafo de estados:

1. Una descripción para etiquetar el *nodo inicial*. Esta descripción estará formada por alguna estructura de datos que nos permita modelar el estado inicial.
2. Las funciones para transformar las descripciones de los estados (que representan estados del mundo) en las descripciones de los estados resultantes al aplicar las funciones.

Normalmente, a estas funciones se les denomina *operadores*. En nuestros ejemplos sobre agentes, los operadores son modelos de los efectos de las acciones. Cuando aplicamos un operador a un nodo, se genera uno de sus nodos sucesores.

3. Una *condición de éxito*, que puede ser una función booleana, que se aplica a las descripciones de los estados, o una lista con las descripciones de estados que se corresponden con los estados objetivos.

En este capítulo y en el siguiente examinaremos dos tipos de procesos de búsqueda. El primer tipo corresponde a procesos de búsqueda en los que no se dispone de información específica del problema para establecer preferencias dentro del espacio de estados en lo que concierne a la búsqueda del camino al objetivo. Este tipo de procesos se denominan *búsquedas a ciegas*. En el otro tipo, nos encontraremos con procesos de búsqueda que disponen de información específica sobre el problema, con lo que se puede mejorar la eficiencia del proceso de búsqueda. Este tipo de procesos se denominan *búsquedas heurísticas*¹. En este capítulo analizaremos el primer tipo, mientras que las búsquedas heurísticas las veremos en el capítulo siguiente.

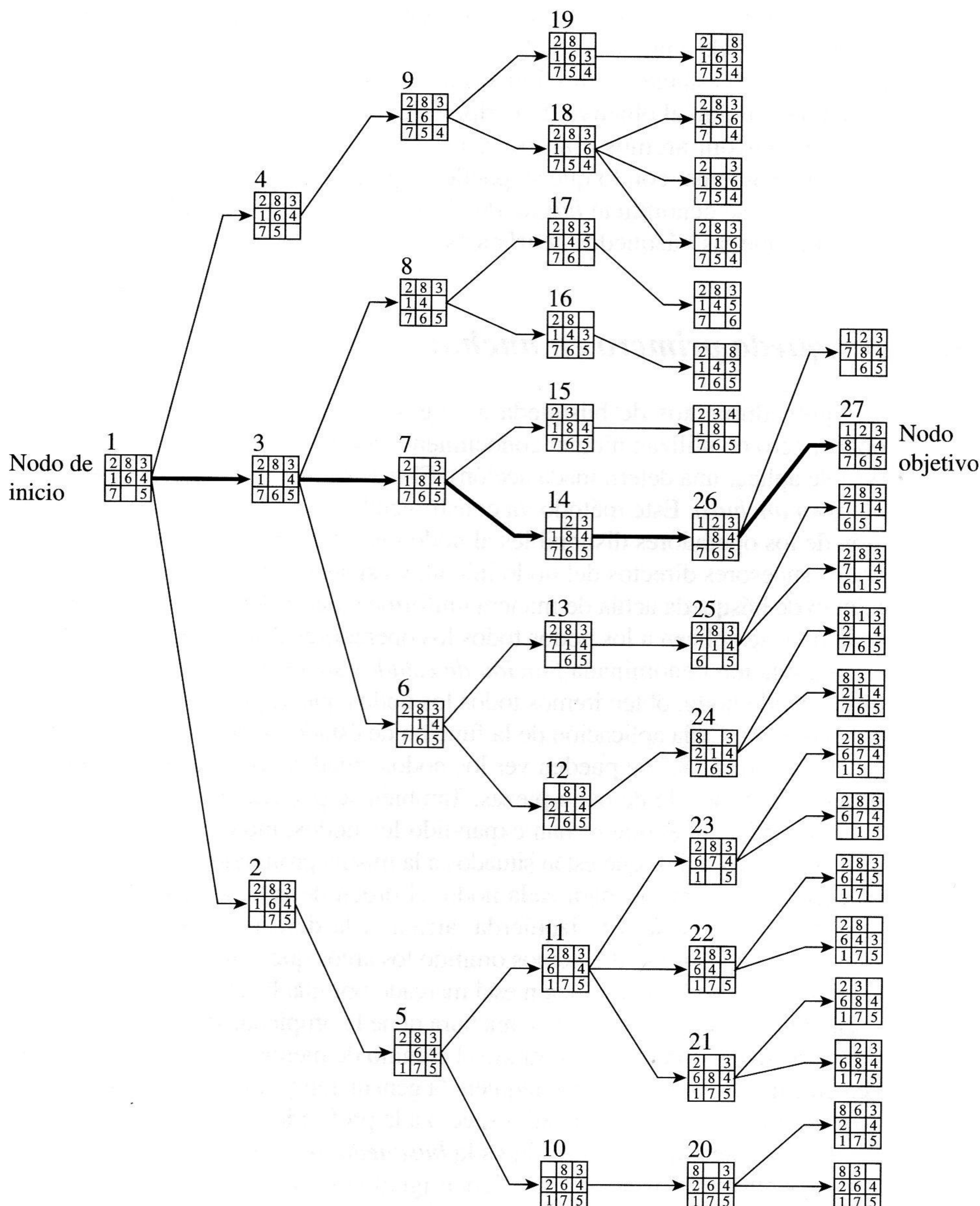
8.3. Búsqueda primero en anchura

Los procedimientos de búsqueda a ciegas operan aplicando los operadores disponibles a los nodos, pero no utilizan ningún conocimiento sobre el dominio del problema (sólo saben cuándo es posible aplicar una determinada acción). El más simple de estos procedimientos es la *búsqueda primero en anchura*. Este método va construyendo un grafo de estados explícito mediante la aplicación de los operadores disponibles al nodo inicial, después aplica los operadores disponibles a los nodos sucesores directos del nodo inicial, y así sucesivamente. Como podemos ver, este procedimiento de búsqueda actúa de manera uniforme a partir del nodo inicial. Teniendo en cuenta que en cada paso se aplican a los nodos todos los operadores disponibles, resulta más eficiente agruparlos en una función denominada *función de estados sucesores*. Cuando apliquemos esta función a un determinado nodo, obtendremos todos los nodos que se producirían al aplicar todos los operadores a dicho nodo. Cada aplicación de la función de estados sucesores se denomina *expansión* del nodo.

En la Figura 8.2 se pueden ver los nodos creados por la búsqueda primero en anchura para el problema del puzzle de ocho piezas. También se pueden apreciar los nodos inicial y objetivo así como el orden en el que se han expandido los nodos, indicado por el número que está próximo a cada nodo. Los nodos que están situados a la misma profundidad se van expandiendo según un orden prefijado de antemano; para cada nodo, el orden de aplicación de los operadores es el siguiente: mover la celda vacía a la izquierda, arriba, a la derecha y abajo. Aunque cada movimiento es reversible, en la Figura 8.2 hemos omitido los arcos que van de los sucesores a sus predecesores. El camino que conforma la solución está marcado por una línea gruesa más oscura. Como veremos más adelante, la búsqueda primero en anchura tiene la propiedad de que una vez que se ha encontrado el nodo objetivo, habremos encontrado el camino de menor longitud. Sin embargo, el principal inconveniente de este método es que requiere la generación y almacenamiento de todo el árbol, cuyo tamaño crece de manera exponencial respecto a la profundidad del nodo objetivo menos profundo.

Una variante de este método es la *búsqueda de coste uniforme* [Dijkstra, 1959], en la que se van expandiendo los nodos que tienen igual coste en vez de expandir los nodos que están a la misma profundidad. Si los costes de todos los arcos del grafo son idénticos, la búsqueda de coste uniforme equivale a una búsqueda primero en anchura. Evidentemente, la búsqueda de coste uniforme se puede considerar como un caso especial de búsqueda heurística, que será descrita en el siguiente capítulo. Este pequeño análisis sobre la búsqueda primero en anchura y la búsqueda de coste uniforme nos pueden dar una idea general sobre los métodos de búsqueda, pero habría que profundizar más en los aspectos técnicos para cubrir los casos en los que el proceso de expansión produzca nodos que ya han sido analizados por el proceso de búsqueda. Dejaremos este análisis para el siguiente capítulo, una vez que tratemos procedimientos de búsqueda más generales.

¹ La palabra *heurística* proviene del griego *heuriskein*, que significa descubrir. De la misma palabra se deriva la exclamación ¡*Eureka!*!

**Figura 8.2.**

Búsqueda primero en anchura para el puzzle de ocho piezas.

8.4. Búsqueda primero en profundidad o búsqueda con vuelta atrás

En este proceso de búsqueda se genera sólo un sucesor del nodo en cada paso (en el primer paso se tratará de un sucesor del nodo inicial); es decir, cada vez que obtenemos un nuevo sucesor, se le aplica a éste un operador y se obtiene un nuevo sucesor, y así sucesivamente. En cada nodo se tiene que dejar una marca que indique, en caso de ser necesario, qué operadores

adicionales se pueden utilizar y especificar el orden en que deben ser aplicados. Para evitar que el proceso de búsqueda continúe este descenso sin detenerse, se establece una profundidad límite. Una vez que se alcance esta *profundidad límite*, la búsqueda detiene el proceso de generación de sucesores (evidentemente, esto significa que se supone que al menos existe un nodo objetivo antes de dicho límite). Este límite nos permite desechar aquellas partes del grafo en las que se supone que no encontraremos un nodo objetivo lo suficientemente cercano al nodo inicial.

Para ilustrar este ejemplo utilizaremos el problema del puzzle de ocho piezas con una profundidad límite de 5. Volveremos a utilizar los operadores que mueven el espacio en blanco hacia la izquierda, arriba, a la derecha y abajo, y volveremos a omitir los arcos que van desde los sucesores a sus predecesores. En la Figura 8.3a se puede ver la generación de los primeros nodos. El número que está a la izquierda de cada nodo indica el orden en que ha sido generado. También se pueden ver los nodos que no han sido totalmente expandidos, indicados con arcos que parten del nodo. Como podemos ver en la figura, con el nodo 5 hemos alcanzado la profundidad límite sin haber alcanzado el nodo objetivo, con lo que nos vemos obligados a retornar a un nodo inmediatamente anterior que todavía admite nuevas expansiones. En nuestro caso, volveríamos a considerar el nodo 4 y generar un nuevo sucesor, el nodo 6 (Fig. 8.3b). Obviamente, podemos desechar el nodo 5, ya que no generaremos más sucesores a partir de él. Sin embargo, con el nodo 6 volvemos a alcanzar la profundidad límite sin llegar al nodo objetivo, con lo que otra vez tenemos que retornar a un nodo inmediatamente superior que admite más expansiones, es decir, al nodo 2, y generar un nuevo sucesor, el nodo 7 —desechando nuevamente el nodo 3 y todos sus sucesores, ya que no volveremos a generar nuevos sucesores de él—. Este retorno al nodo 2 es un ejemplo de *vuelta atrás cronológica*. Siguiendo por este camino, volveremos a alcanzar la profundidad límite (grafo de la Fig. 8.3c) sin alcanzar el nodo objetivo, con lo que habrá que generar un nuevo nodo sucesor del nodo 8, que tampoco es el nodo objetivo. Por tanto, podemos desechar el nodo 8 y todos sus sucesores y retornar al nodo 7, volviendo a generar un nuevo sucesor. En la Figura 8.4 se puede ver el grafo que resulta al aplicar este proceso.

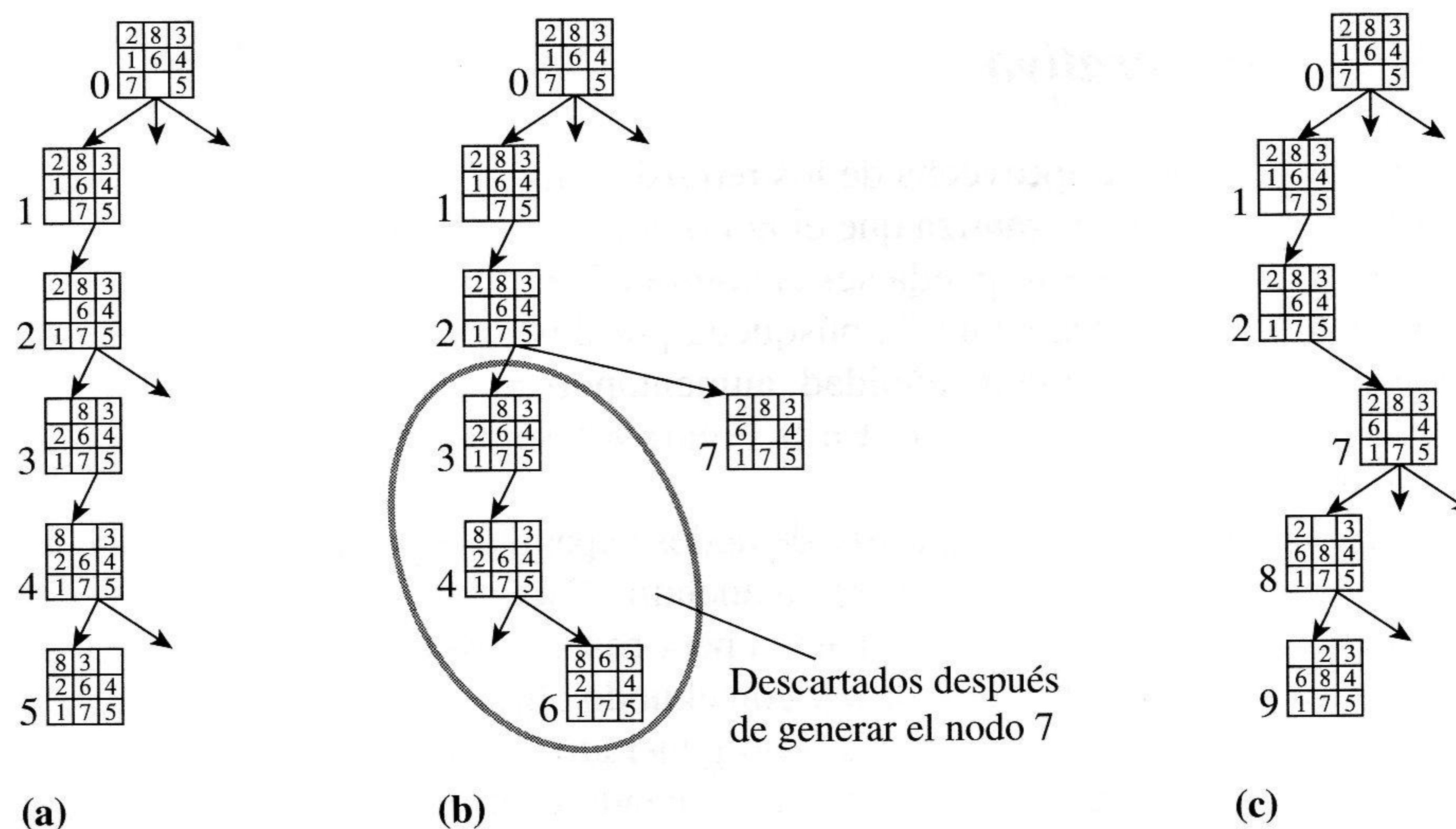
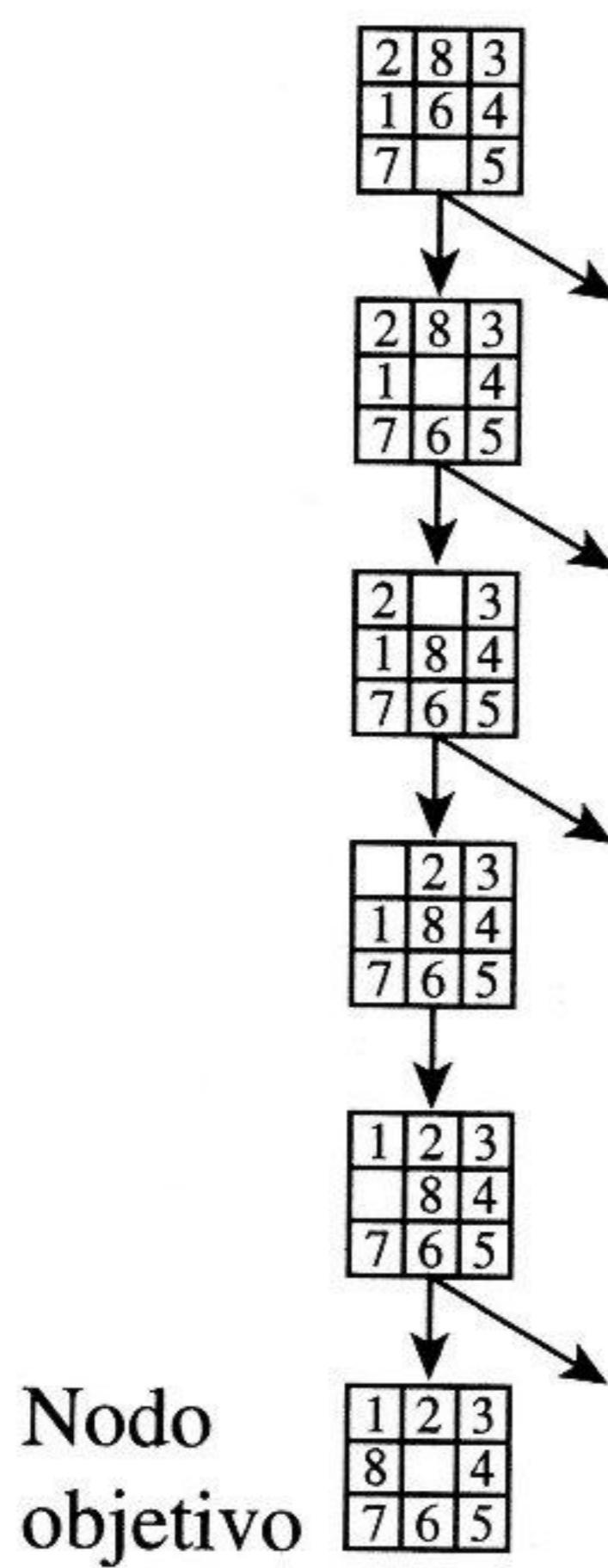


Figura 8.3.

Generación de los primeros nodos en la búsqueda primero en profundidad.

**Figura 8.4.**

Grafo obtenido una vez que se ha alcanzado el objetivo por la búsqueda primero en profundidad.

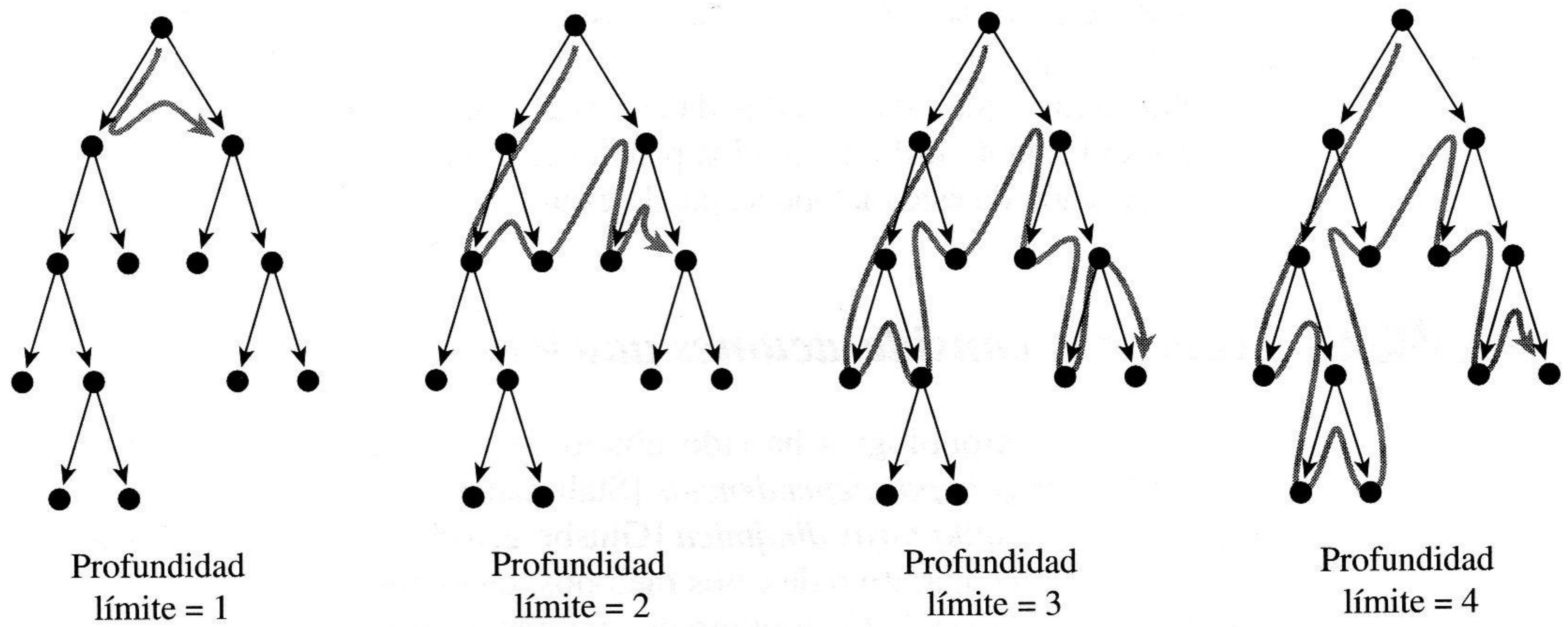
Como podemos apreciar, para la búsqueda primero en profundidad sólo se necesita almacenar la parte del árbol de búsqueda que contiene el camino que está siendo explorado, además de las trazas de los nodos que no han sido totalmente expandidos. Por tanto, el coste en memoria de este proceso crece de forma lineal con la profundidad límite. Sin embargo, una desventaja de este proceso es que una vez que hayamos alcanzado el nodo objetivo, no podemos asegurar que sea el nodo objetivo situado a menor profundidad. Otra desventaja es que, probablemente, tendremos que explorar una gran parte del grafo de estados, incluso para encontrar un nodo objetivo que esté a poca profundidad y que sea descendiente de uno de los últimos nodos en ser expandido.

8.5. Descenso iterativo

Una técnica que se aprovecha de los requisitos lineales de memoria de la búsqueda primero en profundidad, y que garantiza que el nodo objetivo encontrado es el más cercano al nodo inicial (siempre y cuando éste pueda ser encontrado), es el descenso iterativo [Korf, 1985; Stickel y Tyson, 1985]. Básicamente, la búsqueda por descenso iterativo consiste en realizar sucesivas búsquedas primero en profundidad, aumentando en cada paso la profundidad límite, hasta que se encuentre el nodo objetivo. En la Figura 8.5 se puede apreciar cómo opera el *descenso iterativo*.

Sorprendentemente, el número de nodos expandidos por este proceso no es mucho mayor que en el caso de la búsqueda primero en anchura. Para demostrar esta afirmación, vamos a calcular el número de nodos expandidos para el peor caso, considerando un árbol de búsqueda con un factor de ramificación uniforme de b y con el nodo objetivo más cercano a la raíz situado a una profundidad d , siendo, además, el último generado a esa profundidad. Con estos datos, una cota máxima para el número total de nodos expandidos por la búsqueda primero en anchura es:

$$N_{bf} = 1 + b + b^2 + \dots + b^d = \frac{b^{d+1} - 1}{b - 1}$$

**Figura 8.5.**

Etapas de la búsqueda por descenso iterativo.

Para calcular el número total de nodos expandidos en el proceso de descenso iterativo, primero necesitamos saber el número de nodos expandidos en una búsqueda primero en profundidad completa hasta el nivel j , que viene dado por la siguiente expresión:

$$N_{df_j} = \frac{b^{j+1} - 1}{b - 1}$$

En el peor caso, la búsqueda por descenso iterativo completará todas las búsquedas primero en profundidad variando la profundidad límite entre 1 y d . Teniendo en cuenta esto, el número total de nodos expandidos vendrá dado por:

$$\begin{aligned} N_{id} &= \sum_{j=0}^d \frac{b^{j+1} - 1}{b - 1} \\ &= \frac{1}{b-1} \left[b \left(\sum_{j=0}^d b^j \right) - \sum_{j=0}^d 1 \right] \\ &= \frac{1}{b-1} \left[b \left(\frac{b^{d+1} - 1}{b - 1} \right) - (d + 1) \right] \end{aligned}$$

Simplificando la expresión anterior, la cota máxima para el número total de nodos expandidos en un descenso iterativo para un nodo objetivo de profundidad d es:

$$N_{id} = \frac{b^{d+2} - 2b - bd + d + 1}{(b - 1)^2}$$

Teniendo en cuenta que para valores grandes de d el cociente N_{id}/N_{bf} tiende a $b/(b - 1)$, considerando un factor de ramificación de 10 y que los nodos objetivos sean profundos, el proceso de

descenso iterativo sólo expandirá un 11 por 100 más de nodos que en el caso de una búsqueda primero en anchura.

Otra técnica que está relacionada con ésta es la búsqueda por *recorrido iterativo en anchura*, que resulta bastante útil en aquellos problemas con un número considerable de nodos objetivos. Una descripción de este método se puede examinar en [Ginsberg y Harvey, 1992, y Harvey, 1994].

8.6.

Lecturas y consideraciones adicionales

La vuelta atrás cronológica ha sido objeto de varias mejoras entre las que podemos citar la *vuelta atrás dirigida por dependencias* [Stallman y Sussman, 1977], el *salto hacia atrás* [Gaschnig, 1979] y la *vuelta atrás dinámica* [Ginsberg, 1993]. En este último artículo se puede encontrar un estudio comparativo de estos métodos, en el que quedan de manifiesto las ventajas de la vuelta atrás dinámica. Este conjunto de mejoras del esquema clásico de vuelta atrás es muy utilizado en los problemas de satisfacción de restricciones, que serán analizados en el Capítulo 11.

Ejercicios

8.1. En el denominado problema de «las jarras de agua» se dispone de dos jarras, una de 3 litros, a la que llamaremos *tres*, y otra de 4 litros, a la que llamaremos *cuatro*. Inicialmente, ambas jarras están vacías. Disponemos de un grifo, *G*, del cual podemos coger agua para llenar las jarras, además de un desagüe, *D*, que podemos utilizar para vaciar (total o parcialmente) las jarras, y se puede pasar agua de una jarra a otra. No disponemos de ningún dispositivo para realizar medidas. El objetivo del problema es encontrar un conjunto de operaciones que consigan dejar exactamente 2 litros de agua en la jarra *cuatro*. [Existe una solución: *a*) llenar la jarra *tres* con el agua del grifo; *b*) llenar la jarra *cuatro* con el agua que hay en la jarra *tres*; *c*) volver a llenar la jarra *tres* con el agua del grifo; *d*) rellenar la jarra *cuatro* con el agua de la *tres*; *e*) vaciar la jarra *cuatro*, y *f*) volver a llenar la jarra *cuatro* con el agua que queda en la jarra *tres*, que es exactamente 2 litros]. Se pide que:

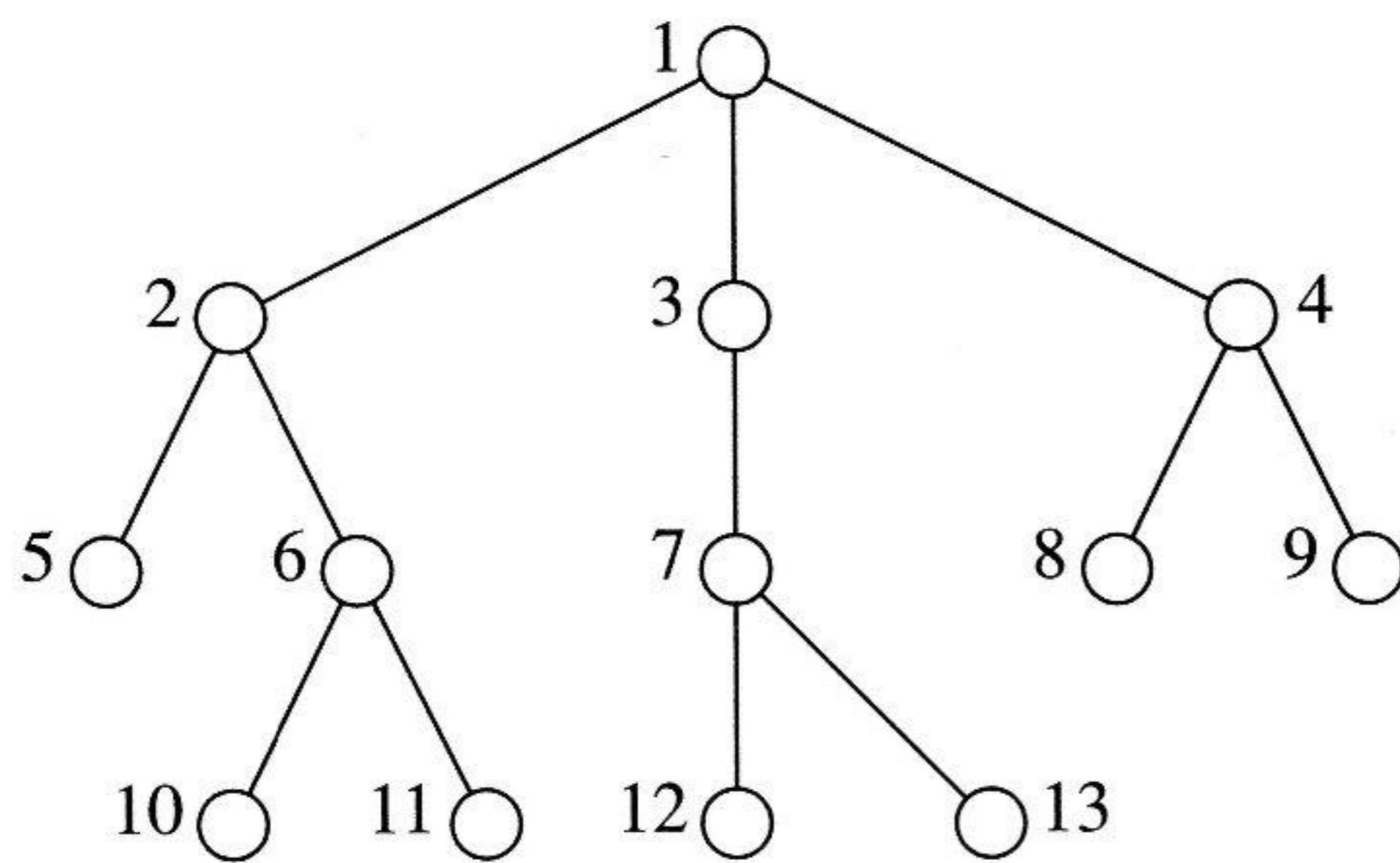
1. Defina un espacio de estados para resolver el problema mediante un procedimiento de búsqueda:

- a.* Defina la descripción icónica de los estados mediante una estructura de datos.
- b.* Defina la condición de éxito sobre dicha estructura de datos.
- c.* Defina el conjunto de operadores que se pueden aplicar a cada estado, dando una explicación detallada de los efectos de los mismos.

2. Dibuje un grafo con todos los nodos *distintos* que están a una distancia de tres movimientos del nodo inicial, etique cada nodo con el estado que representa y cada arco con el operador aplicado. Dibuje el camino hacia la solución.

8.2. Liste el orden en el que son visitados los nodos del árbol de la figura para cada una de las siguientes estrategias de búsqueda (eliendo siempre el nodo más a la izquierda):

1. Búsqueda primero en profundidad.
2. Descenso iterativo (aumentando en 1 la profundidad límite en cada iteración).
3. Búsqueda primero en anchura.



8.3. Consideremos un árbol finito de profundidad d y con un factor de ramificación b (un árbol que sólo tiene el nodo raíz tiene profundidad 0; un árbol compuesto por el nodo raíz y sus b sucesores directos tiene profundidad 1, etc.). Supongamos que el nodo objetivo menos profundo está a una profundidad $g \leq d$.

1. ¿Cuál es el número *mínimo* de nodos generados por la *búsqueda primero en profundidad* con una profundidad límite d ? ¿Y el *máximo*?
2. ¿Cuál es el número *mínimo* de nodos generados por la *búsqueda primero en anchura*? ¿Y el *máximo*?
3. ¿Cuál es el número *mínimo* de nodos generados por el *descenso iterativo*? ¿Y el *máximo*? (considere que empezamos con una profundidad límite inicial de 1 y la vamos incrementando una unidad en cada iteración).

8.4. Supongamos que mediante la búsqueda primero en anchura se puedan generar 10.000 nodos por segundo y que para almacenar cada nodo son necesarios 100 bytes. Con estos datos ¿cuáles son los requisitos de tiempo y memoria para una búsqueda primero en anchura aplicada a un árbol de búsqueda de profundidad d y un factor de ramificación de 5? Muestre los resultados del análisis en una tabla (se pueden usar números aproximados cuando resulte más fácil expresar el tiempo en horas, días, meses, años, etc.)

8.5. Supongamos que estamos aplicando un procedimiento de búsqueda sobre un árbol con factor de ramificación 5. Sin embargo, en realidad no sabemos que estamos trabajando sobre un árbol, con lo que tenemos que comprobar cada nuevo estado para ver si hemos alcanzado un estado ya analizado. ¿Cuántas comprobaciones tendremos que hacer en un árbol de búsqueda de profundidad d ?