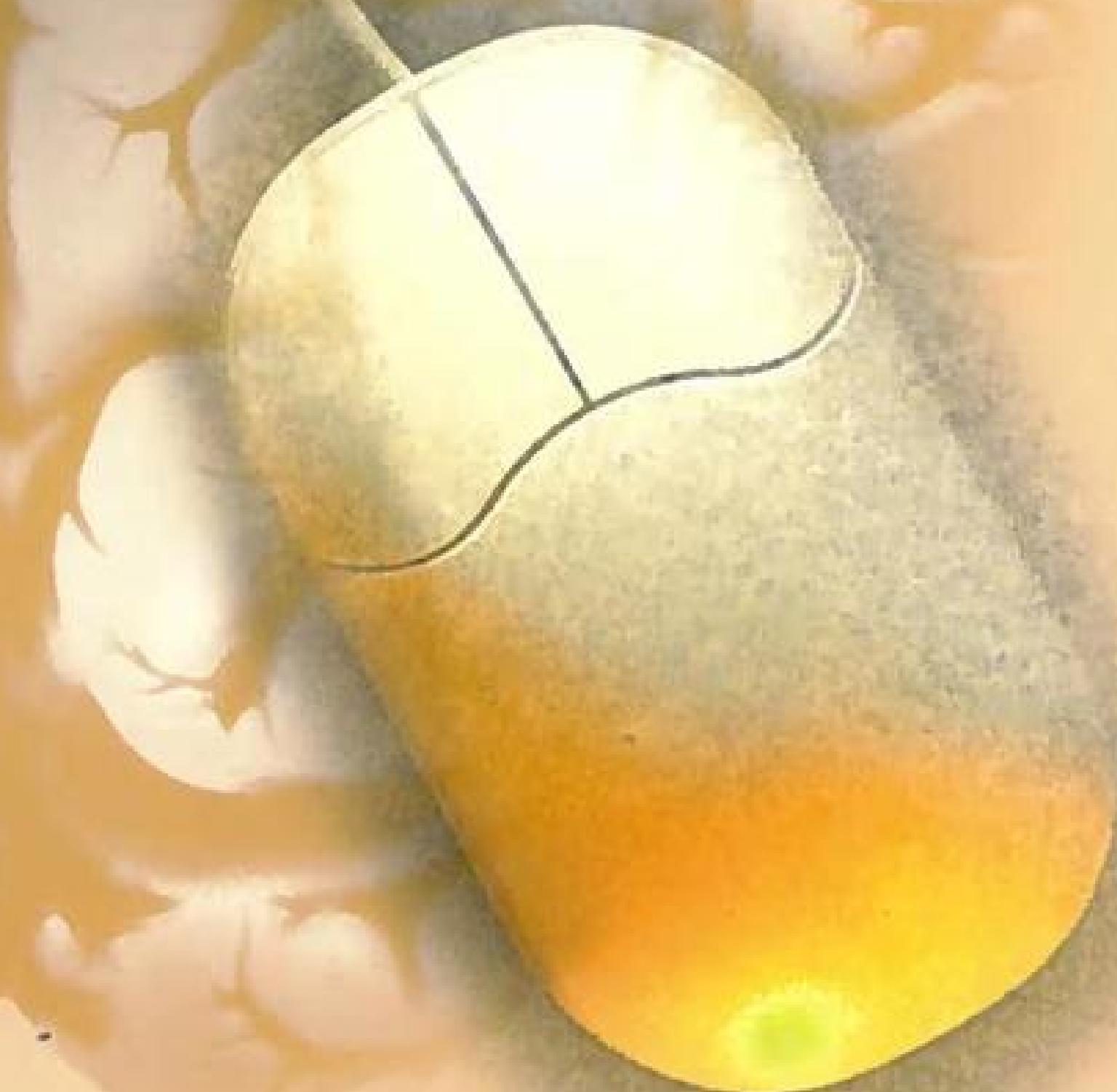


# Inteligencia artificial

Modelos, Técnicas  
y Áreas de Aplicación

Francisco Escolano Ruiz  
Miguel Ángel Cazorla Quevedo  
M<sup>a</sup> Isabel Alfonso Galipienso  
Otto Colomina Pardo  
Miguel Ángel Lozano Ortega



# Paraninfo

## INTELIGENCIA ARTIFICIAL. MODELOS, TÉCNICAS Y ÁREAS DE APLICACIÓN

© FRANCISCO ASCOLANO, MIGUEL ÁNGEL CAZORLA, M<sup>ª</sup> ISABEL ALFONSO, OTTO COLOMINA Y  
MIGUEL ÁNGEL LOZANO

**Gerente Editorial Área Universitaria:**  
Andrés Otero Reguera

**Diseño de cubierta:**  
Montytexto

**Editoras de Producción:**  
Clara M<sup>ª</sup> de la Fuente Rojo  
Consuelo García Asensio

**Impresión:**  
Graficas Rogar  
Pto Ind. Alparreche  
Navalcarnero (Madrid)

**Producción Industrial:**  
Susana Pavón Sánchez

COPYRIGHT © 2003 International  
Thomson Ediciones Spain  
Paraninfo, S.A.

Magallanes, 25;  
28015 Madrid, ESPAÑA  
Teléfono: 902 995 240  
Fax: 914 456 218  
[clientes@paraninfo.es](mailto:clientes@paraninfo.es)  
[www.paraninfo.es](http://www.paraninfo.es)

Impreso en España  
Printed in Spain

ISBN: 84-9732-183-9  
Depósito legal: M-44.165-2003

(102/68/48)

Reservados los derechos para todos los países de lengua española. De conformidad con lo dispuesto en el artículo 270 del Código Penal vigente, podrán ser castigados con penas de multa y privación de libertad quienes reprodujeren o plagiaren, en todo o en parte, una obra literaria, artística o científica fijada en cualquier tipo de soporte sin la preceptiva autorización. Ninguna parte de esta publicación, incluido el diseño de la cubierta, puede ser reproducida, almacenada o transmitida de ninguna forma, ni por ningún medio, sea éste electrónico, químico, mecánico, electro-óptico, grabación, fotocopia o cualquier otro, sin la previa autorización escrita por parte de la Editorial.

# Índice general

PRÓLOGO	IX
INTRODUCCIÓN	XI
I. TÉCNICAS FUNDAMENTALES	1
1. INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL	3
1.1. Definición de Inteligencia Artificial . . . . .	3
1.2. Áreas de aplicación . . . . .	4
1.3. Origen y evolución de la Inteligencia Artificial . . . . .	4
Bases de la IA moderna . . . . .	4
Definición del campo: conferencia de Dartmouth y los años dorados . . . . .	5
Las conquistas de los micro-mundos . . . . .	6
Años de crítica y madurez: los difíciles años setenta . . . . .	6
Etapa de expansión: los años ochenta . . . . .	7
Estado actual . . . . .	7
2. BÚSQUEDA HEURÍSTICA	9
2.1. <i>Simulated Annealing</i> . . . . .	9
Problemas combinatoriales . . . . .	9
Relajación estocástica . . . . .	10
2.2. Algoritmo $A^*$ . . . . .	17
Concepto de heurística . . . . .	17
Algoritmo $A^*$ sobre grafos . . . . .	21
Admisibilidad de $A^*$ . . . . .	23
Variantes de $A^*$ . . . . .	27
2.3. Juegos: $\alpha - \beta$ y MTD-f . . . . .	30
Búsquedas MINIMAX y $\alpha - \beta$ . . . . .	30
Algoritmo MTD-f . . . . .	36

<b>2.4. Satisfacción de restricciones . . . . .</b>	<b>41</b>
Grafos de restricciones . . . . .	41
<i>Backtracking</i> y <i>backjumping</i> . . . . .	44
Propagación de restricciones . . . . .	47
<b>2.5. Ejercicios . . . . .</b>	<b>53</b>
<b>2.6. Referencias bibliográficas . . . . .</b>	<b>54</b>
<b>3. REPRESENTACIÓN E INFERENCIA . . . . .</b>	<b>57</b>
<b>3.1. Representación mediante lógica . . . . .</b>	<b>57</b>
La inferencia automática en lógica . . . . .	59
Sistemas de programación lógica: el lenguaje PROLOG . . . . .	67
<b>3.2. Sistemas de reglas . . . . .</b>	<b>69</b>
Inferencia en sistemas de reglas . . . . .	69
CLIPS: un entorno de desarrollo de sistemas basados en reglas . . . . .	71
<b>3.3. Lógica fuzzy . . . . .</b>	<b>74</b>
Introducción . . . . .	74
Conjuntos fuzzy . . . . .	75
Fundamentos de la lógica difusa . . . . .	80
Sistemas expertos difusos . . . . .	84
<b>3.4. Ejercicios . . . . .</b>	<b>88</b>
<b>3.5. Referencias bibliográficas . . . . .</b>	<b>88</b>
<b>4. APRENDIZAJE . . . . .</b>	<b>91</b>
<b>4.1. Redes neuronales . . . . .</b>	<b>91</b>
Perceptrones . . . . .	91
Memorias asociativas . . . . .	105
Mapas auto-organizativos (SOMs) . . . . .	114
<b>4.2. Aprendizaje bayesiano . . . . .</b>	<b>119</b>
Bayes Naïf . . . . .	119
Modelos ocultos de Markov (HMMs) . . . . .	123
<b>4.3. Aprendizaje de árboles de decisión . . . . .</b>	<b>136</b>
<b>4.4. Aprendizaje evolutivo . . . . .</b>	<b>140</b>
De la inspiración biológica al modelo computacional . . . . .	141
Aplicación de algoritmos evolutivos . . . . .	143
Representación de las soluciones . . . . .	147
Operadores de cruce . . . . .	148
Operadores de mutación . . . . .	149
Operadores de selección . . . . .	150
<b>4.5. Ejercicios . . . . .</b>	<b>150</b>

<b>4.6. Referencias bibliográficas . . . . .</b>	<b>152</b>
<b>II. ÁREAS DE APLICACIÓN . . . . .</b>	<b>155</b>
<b>5. PLANIFICACIÓN Y SCHEDULING . . . . .</b>	<b>157</b>
<b>    5.1. Planificación . . . . .</b>	<b>157</b>
Definición de planificación . . . . .	157
Planificación clásica . . . . .	158
Planificación como búsqueda en el espacio de estados . . . . .	164
Planificación como búsqueda en el espacio de planes . . . . .	168
Planificación disyuntiva . . . . .	174
Planificación como satisfacción de restricciones . . . . .	179
<b>    5.2. Scheduling . . . . .</b>	<b>183</b>
Definición de <i>scheduling</i> . . . . .	183
Tipología de problemas de <i>scheduling</i> . . . . .	185
<i>Job-Shop scheduling</i> . Aproximaciones . . . . .	187
Técnicas de optimización . . . . .	188
Técnicas de aproximación basadas en Inteligencia Artificial . . . . .	189
Conceptos sobre redes de restricciones . . . . .	190
Técnicas de satisfacción de restricciones . . . . .	192
Técnicas de clausura: deducción y propagación . . . . .	193
Heurísticas y medidas de textura . . . . .	193
Métodos de búsqueda local . . . . .	194
<b>    5.3. Ejercicios . . . . .</b>	<b>196</b>
<b>    5.4. Referencias bibliográficas . . . . .</b>	<b>196</b>
<b>6. LENGUAJE NATURAL . . . . .</b>	<b>199</b>
<b>    6.1. Introducción . . . . .</b>	<b>201</b>
Pasos en el proceso de análisis . . . . .	201
Problemáticas en el lenguaje . . . . .	205
<b>    6.2. Análisis morfológico . . . . .</b>	<b>206</b>
Características morfológicas . . . . .	207
Algoritmo de análisis morfológico . . . . .	208
<b>    6.3. Análisis sintáctico . . . . .</b>	<b>208</b>
Gramáticas y árboles sintácticos . . . . .	209
Tratamiento de la ambigüedad . . . . .	211
Análisis sintáctico descendente . . . . .	213
Análisis ascendente . . . . .	216
Redes de transición . . . . .	220
<b>    6.4. Comprensión del lenguaje natural . . . . .</b>	<b>225</b>

<b>6.5. Análisis semántico . . . . .</b>	<b>226</b>
Forma lógica . . . . .	227
Composicionalidad . . . . .	229
Construcción de la forma lógica . . . . .	230
Ambigüedad en el análisis semántico . . . . .	232
Gramáticas semánticas . . . . .	238
<b>6.6. Análisis pragmático e integración del discurso . . . . .</b>	<b>239</b>
Integración del discurso . . . . .	239
Comprensión del lenguaje natural . . . . .	241
<b>6.7. Ejercicios . . . . .</b>	<b>241</b>
<b>6.8. Referencias bibliográficas . . . . .</b>	<b>244</b>
<b>7. VISIÓN ARTIFICIAL . . . . .</b>	<b>245</b>
<b>7.1. Restauración de imágenes . . . . .</b>	<b>245</b>
Formulación del problema . . . . .	245
Restauración estocástica . . . . .	248
<b>7.2. Extracción de características . . . . .</b>	<b>250</b>
Características de contorno . . . . .	250
Características de región . . . . .	261
<b>7.3. Segmentación . . . . .</b>	<b>277</b>
Segmentación de contornos . . . . .	277
Segmentación de regiones . . . . .	282
<b>7.4. Reconocimiento de objetos . . . . .</b>	<b>289</b>
Métodos de <i>bitmap</i> . . . . .	289
Métodos de búsqueda . . . . .	294
<b>7.5. Ejercicios . . . . .</b>	<b>301</b>
<b>7.6. Referencias bibliográficas . . . . .</b>	<b>304</b>
<b>8. ROBÓTICA . . . . .</b>	<b>307</b>
<b>8.1. Introducción a los sistemas robóticos . . . . .</b>	<b>307</b>
¿Qué es un robot? . . . . .	307
Revisión histórica . . . . .	308
El paradigma reactivo . . . . .	310
<b>8.2. Sensores . . . . .</b>	<b>313</b>
Medición de la posición del robot . . . . .	313
Sensores de localización . . . . .	314
Sensores de proximidad . . . . .	315
<b>8.3. Visión para robots . . . . .</b>	<b>319</b>
Detección de características . . . . .	319

Obtención de la profundidad mediante visión estéreo . . . . .	320
<b>8.4. Planificación de trayectorias . . . . .</b>	<b>327</b>
Representación del espacio de configuración . . . . .	327
Geometría del robot . . . . .	329
Planificación basándose en grafos . . . . .	334
Campos de potencial . . . . .	336
Navegación topológica . . . . .	340
<b>8.5. Navegación evitando obstáculos . . . . .</b>	<b>345</b>
El enfoque de la ventana dinámica . . . . .	346
<b>8.6. Localización y construcción de mapas . . . . .</b>	<b>350</b>
Localización del robot . . . . .	350
Construcción de mapas . . . . .	352
<b>8.7. Áreas de aplicación . . . . .</b>	<b>356</b>
Guías robóticos . . . . .	356
Robocup . . . . .	357
Sistemas de ayuda a la conducción . . . . .	358
<b>8.8. Ejercicios . . . . .</b>	<b>359</b>
<b>8.9. Referencias bibliográficas . . . . .</b>	<b>360</b>
<b>BIBLIOGRAFÍA</b>	<b>361</b>
<b>ÍNDICE ALFABÉTICO</b>	<b>367</b>



# Prólogo

¿Qué es la Inteligencia Artificial?, ¿por qué el nombre “inteligencia”?-, ¿podrán las máquinas llegar a ser alguna vez más inteligentes que los humanos?, ¿se puede explicar la mente humana en términos idénticos a los de las computadoras?, ¿y viceversa?, ¿en qué afecta la Inteligencia Artificial a nuestra vida?, ¿cuáles son sus aplicaciones y logros actuales?, ¿cuáles son los límites de la Inteligencia Artificial?, etc.

Éstas son algunas de las preguntas, en parte inquietantes, que puede hacerse cualquier persona frente a la Inteligencia Artificial. Sin embargo, a muchas de estas preguntas no encontraremos respuesta en este libro porque, hoy por hoy, los que trabajamos en el entorno de la Inteligencia Artificial estamos más preocupados por su desarrollo, divulgación, aplicación y la resolución de los retos que nos plantea, que por su expectativas, promesas o futuribles aún todavía lejanos.

Por ello, es personalmente un honor poder incluir unas frases al principio de este libro. Es de los pocos, demasiado escasos más bien, libros sobre Inteligencia Artificial escritos originalmente por autores de habla hispana. Además, merece destacarse su claridad, exposición pedagógica y cubrimiento de los temas contenidos en la obra.

Hay que reconocer el esfuerzo de los autores en la redacción del libro. Este reconocimiento se hace aún más sincero por la especial y cercana relación que me une a algunos de sus autores. Considero que es un libro dirigido tanto a los que se introducen de nuevas en el fascinante mundo de la Inteligencia Artificial, permitiéndoles conocer algunos de sus modelos, técnicas y aplicaciones más destacadas; como a los que ya conocemos y trabajamos en sus diversos campos, pudiéndonos servir como un texto que amplíe nuestros conocimientos sobre las áreas específicas que trata. En conjunto, con una exposición clara y pedagógica, complementada con múltiples referencias y ejemplos, constituye una útil referencia para nuestro trabajo.

La Inteligencia Artificial, definida como “el estudio de cómo programar computadoras que posean la facultad de hacer aquello que la mente humana puede realizar” (Minsky), toma un sentido científico viable durante la segunda mitad del siglo pasado, como resultado directo de la confluencia de diversas corrientes intelectuales desarrolladas sobre los cimientos formales de la lógica y la matemática discreta, e impulsadas por el desarrollo de los computadores digitales. Particularmente, la Inteligencia Artificial supone un serio esfuerzo por entender la complejidad de la conducta humana en términos de proceso de información.

Y mucho se ha logrado desde aquellos ya lejanos inicios. La Inteligencia Artificial es un área extraordinariamente vital, amplia y multidisciplinar. El comportamiento inteligente humano, como el que la Inteligencia Artificial trata de emular y/o simular, presenta complejos aspectos “cognitivos”, “perceptivos”, “heurísticos”, “sociales”, “colaborativos”, etc., hasta futuribles aspectos “emocionales”. Entre todos ellos, se encuentra una amplia variedad de orientaciones, considerando tanto aspectos propios del pensamiento humano, como aspectos relacionados con su comportamiento y conducta. Cuando examinamos la evolución de la IA, en su reciente medio siglo de historia, observamos una transición desde las teorías y sistemas embrionarios iniciales a los adaptables, flexibles, robustos y

amigables entornos actuales, basados en una amplia variedad de teorías, lógicas, modelos, técnicas y aproximaciones desde la ingeniería. El desarrollo tecnológico y el progreso en campos afines también tendrán algo que decir en el futuro. Mediante un análisis de los sistemas de Inteligencia Artificial actuales y del modo en que éstos pueden ser ampliados, estaremos en disposición de identificar una diversidad de interrogantes cuya respuesta nos irá aproximando a sistemas inteligentes de propósito más general.

En definitiva, un área en la que hay mucho hecho, pero mucho más por hacer. Un área en la que los retos son difíciles, extraordinariamente complejos, pero también muy prometedores; con una gran vitalidad científica e impacto en nuestra vida. Todo esto forma parte de los motivos, ilusión y esfuerzo continuado que ponemos en el desarrollo, aplicación y transmisión de su conocimiento.

En todo este contexto, por lo que supone como herramienta de divulgación y comprensión, por llenar un hueco excesivamente amplio de textos sobre Inteligencia Artificial desde nuestro entorno más cercano, y por el resultado obtenido de un gran esfuerzo y dedicación, sólo cabe dar la bienvenida, felicitación y reconocimiento a los autores por el trabajo realizado y presentado en esta obra.

Federico Barber Sanchís  
*Presidente de AEPIA<sup>1</sup>*  
Valencia 2003

---

<sup>1</sup> Asociación Española para la Inteligencia Artificial.

# Introducción

Nuestro propósito al escribir este libro sobre Inteligencia Artificial ha sido aportar un texto que sirva de base para impartir IA en la carrera de Informática. En primer lugar, hemos considerado las características propias de la docencia de esta materia, ya que las asignaturas de IA se suelen ubicar en los cursos superiores, de tal forma que los contenidos más asentados forman parte de la troncalidad, mientras que los más avanzados suelen ser el motivo de diversas asignaturas optativas. Por esta razón, nuestro texto se ajusta a un esquema en el que introducimos por un lado los tópicos que entendemos forman un núcleo común en todas las asignaturas introductorias a IA (búsqueda, representación del conocimiento, inferencia y aprendizaje) y por otro lado introducimos tópicos avanzados (planificación y *scheduling*, procesamiento del lenguaje natural, visión artificial y robótica) que se suelen impartir en asignaturas optativas.

Creemos sinceramente que esta estructuración de contenidos permite aportar flexibilidad a la hora de diseñar un curso introductorio a la IA y que, por otro lado, introduce con una extensión razonable los temas típicamente avanzados. Por ello, no nos hemos limitado a introducir tímidamente estos temas avanzados sino que hemos dedicado espacio suficiente en el libro para tratarlos de tal forma que el lector comprenda su complejidad y conozca las aproximaciones más actuales. No se trata, pues, de dejar constancia de que existen campos como la visión artificial o la robótica en donde las técnicas de IA tienen mucho que decir, sino de sugerir que el tratamiento de estas problemáticas supone no sólo un buen campo de prácticas para el desarrollo de la IA sino también un cúmulo de nuevas ideas, técnicas y aproximaciones a problemas más generales.

Profundizando un poco más en la estructura del libro, la primera parte de *Técnicas fundamentales* comienza con una revisión histórica de la IA. A continuación, tratamos los temas relacionados con **búsqueda inteligente** o heurística: *simulated annealing* en búsqueda estocástica,  $A^*$  y sus variantes en búsqueda heurística,  $\alpha - \beta$  y MTD-f en juegos, y *backtracking*, *backjumping* y técnicas de propagación en problemas de satisfacción de restricciones.

En la parte de **representación e inferencia** cubrimos la interrelación entre la representación del conocimiento, y la inferencia y razonamiento. Así, revisamos en primer lugar la representación mediante la lógica y las técnicas de inferencia clausal tipo PROLOG. Después estudiamos los sistemas de reglas y el encadenamiento inferencial tipo CLIPS y, finalmente, presentamos las representaciones basadas en lógica fuzzy y los sistemas basados en principios, según la nomenclatura de Kosko.

Cerramos la primera mitad del libro con la parte de **aprendizaje**. Iniciamos nuestro recorrido presentando tres tipos de red neural: los perceptrones, las memorias asociativas y los mapas auto-organizativos, ya que en cada uno de ellos destaca una aproximación diferente al concepto de red neural y, al mismo tiempo, de cada uno de ellos se derivan aplicaciones también diferentes. A continuación introducimos los aspectos básicos del aprendizaje bayesiano, revisando con detalle los modelos ocultos de Markov. Seguidamente estudiamos los árboles de decisión y, finalmente, cerramos esta parte del libro examinando las implicaciones de la computación evolutiva en el aprendizaje.

La segunda parte del libro, *Áreas de aplicación*, comienza con el tratamiento de los problemas de **planificación y scheduling**. En la parte de planificación se abordan distintos tipos de estrategia: bús-

queda en el espacio de estados, búsqueda en el espacio de planes, planificación disyuntiva, y búsqueda como satisfacción de restricciones. En el capítulo de *scheduling* nos centramos fundamentalmente en los problemas de *job-shop scheduling*, presentando también diversas aproximaciones como las técnicas de optimización o las de clausura (satisfacción de restricciones).

En el capítulo de **lenguaje natural** presentamos las etapas del procesamiento: análisis morfológico, análisis sintáctico ascendente y descendente, comprensión del lenguaje natural, análisis semántico, análisis pragmático e integración del discurso. El denominador común de los algoritmos presentados en este capítulo es el tratamiento de la ambigüedad en la etapa pertinente. Mostramos las conexiones con tópicos clásicos como la búsqueda, especialmente mediante técnicas de satisfacción de restricciones, y la representación del conocimiento, especialmente la lógica.

La introducción a la **visión artificial** que ubicamos en el capítulo siguiente pretende ser un recorrido de complejidad creciente por todas y cada una de las etapas de la visión: desde la restauración de imágenes, en donde establecemos conexiones con la búsqueda mediante *simulated annealing*, hasta el reconocimiento de objetos, ya sean basados en el *bitmap* o de búsqueda, pasando por la extracción de características, tanto de contorno como de región, y la segmentación de imágenes, incluidos el modelado de textura, los modelos de color y el *clustering*.

Finalmente, cerramos esta segunda parte, y con ello el libro, con una introducción a la **robótica**, especialmente a la robótica móvil, más próxima a la IA. Después de una breve revisión de los sensores y de algunos elementos de visión para robots, se entra en detalle en la resolución de problemas como la planificación de trayectorias, la navegación evitando obstáculos, y la localización y construcción de mapas del entorno.

Como vemos, cada uno de los capítulos del libro trata con cierto detalle una problemática concreta. En la mayoría de los casos, dicho tratamiento está dirigido por un ejemplo motivador que se introduce al principio del capítulo. En torno a dicho ejemplo y a otros que se van incluyendo, se presentan los distintos algoritmos o aproximaciones, así como resultados de simulaciones de estos algoritmos, ya sea utilizando software propio o herramientas disponibles en la web. También presentamos trazas paso a paso, destacando casos especiales de los algoritmos para una mejor comprensión de los mismos. Cada capítulo finaliza con una lista de ejercicios y con una serie de referencias bibliográficas comentadas.

Finalmente, sólo nos resta pedir disculpas y dar las gracias. Pedir disculpas porque, debido a limitaciones de tiempo y espacio, hemos dejado fuera campos de gran interés actual como los agentes inteligentes o la ingeniería del conocimiento. En próximas ediciones o en un segundo volumen intentaremos paliar estas deficiencias. Dar las gracias a todos aquéllos que nos han apoyado a lo largo del proceso de diseñar, escribir y corregir este texto: a nuestros compañeros del *Robot Vision Group*, a nuestro amigo Federico Barber por su magnífico prólogo, a Mari Carmen Roncero, Editora de Desarrollo de Thomson-Paraninfo, por su paciencia y comprensión, y a Andrés Otero, Director Editorial del Área Universitaria de Thomson-Paraninfo, por su confianza y apoyo durante todos estos meses.

# **Parte I**

## **Técnicas fundamentales**



# 1

# Introducción a la Inteligencia Artificial

## CONTENIDO DEL CAPÍTULO

- Sección 1.1: definición de Inteligencia Artificial.
- Sección 1.2: áreas de aplicación.
- Sección 1.3: origen y evolución de la Inteligencia Artificial.

En este capítulo procedemos a la revisión del campo de la Inteligencia Artificial desde un punto de vista histórico. Definiremos el concepto de Inteligencia Artificial, así como la evolución desde su definición en 1956. Comentaremos las perspectivas actuales y las líneas de investigación para terminar detallando las posibles fuentes de información bibliográfica de las que podemos hacer uso.

## 1.1 Definición de Inteligencia Artificial

Antes de definir el término Inteligencia Artificial deberíamos definir qué entendemos por inteligencia. La Real Academia de la Lengua Española nos define la inteligencia como “Potencia intelectual: facultad de conocer, de entender o comprender”. Una vez definido esto, la inteligencia artificial se podría definir de la misma forma pero aplicado a las máquinas. De esta forma, cuando nos dicen que una lavadora es inteligente tendríamos que preguntar qué conocimiento o comprensión tiene dicha lavadora del proceso que realiza. Como podemos observar, este término es muy ambiguo y se han aducido múltiples argumentos a favor y en contra de la inteligencia en las máquinas.

Nosotros vamos a utilizar la definición de inteligencia artificial que pensamos que es más cercana a la realidad. La propuso Marvin Minsky, uno de los pioneros de la IA, y dice así: “La Inteligencia Artificial es la ciencia de construir máquinas para que hagan cosas que, si las hicieran los humanos, requerirían inteligencia”.

Podemos pensar en la IA como en aquella ciencia que incorpora conocimiento a los procesos o actividades para que éstos tengan éxito. Un ejemplo es el ajedrez. Es impensable que un computador evalúe todas las posibles jugadas del ajedrez. En vez de esto, se incorpora conocimiento en el proceso de búsqueda de la mejor jugada en forma de jugadas predefinidas o procedimientos de evaluación “inteligentes”.

## 1.2 Áreas de aplicación

Diversas son las áreas donde la IA se presenta en mayor o menor medida. A continuación se comentan brevemente algunas de estas áreas:

- **Tratamiento de lenguajes naturales** En este campo se puede englobar aplicaciones que realicen traducciones entre idiomas, interfaces hombre-máquina que permitan interrogar una base de datos o dar órdenes a un sistema operativo, etc., de manera que la comunicación sea más amigable al usuario.
- **Sistemas expertos** En esta área están englobados aquellos sistemas donde la experiencia de personal cualificado se incorpora a dichos sistemas para conseguir deducciones más cercanas a la realidad.
- **Robótica** Navegación de robots móviles, control de brazos de robots, ensamblaje de piezas, etc.
- **Problemas de percepción: visión y habla** Reconocimiento de objetos y del habla, detección de defectos en piezas por medio de visión, apoyo en diagnósticos médicos, etc.
- **Aprendizaje** Modelización de conductas para su posterior implantación en computadoras.

## 1.3 Origen y evolución de la Inteligencia Artificial

En esta sección haremos un repaso al origen y la evolución de la Inteligencia Artificial. Lo expuesto aquí es un resumen del libro de Crevier [Crevier, 1996].

### Bases de la IA moderna

La Inteligencia Artificial, definida usualmente como *la ciencia de construir máquinas para que hagan cosas que, si las hicieran los humanos, requerirían inteligencia*, toma un sentido científico viable, como disciplina informática moderna, durante la segunda mitad de este siglo. Es el resultado directo de la confluencia de diversas corrientes intelectuales (Teoría de la Computación, Cibernetica, Teoría de la Información, Procesamiento Simbólico) desarrolladas sobre los cimientos formales de la Lógica y la Matemática Discreta, e impulsadas por el desarrollo de los computadores digitales.

Los primeros esfuerzos en el estudio de la simulación automática de la inteligencia tuvieron lugar en el período de 1945 a 1956. Durante este período las diferencias fundamentales entre el funcionamiento del cerebro humano, los sistemas de retro-alimentación (*feedback*) y los computadores digitales no estaban aún bien definidas. En 1943 Wiener, en colaboración con Rosenblueth y Bigelow, establece las bases de la Cibernetica (control, autorregulación) integrando aproximaciones mecánicas (automatismos), biológicas (regulación natural), fisiológicas (neuronas), formales (Lógica) y de procesamiento de información. McCulloch y Pitts explican formalmente la realización de operaciones lógicas mediante el interconexión neural y su control por realimentación, demostrando que cualquier ley de entrada/salida podría implementarse con una red neuronal. Seis años más tarde, en 1949, Hebb sugerirá mecanismos de aprendizaje. Acababa de nacer el paradigma conexiónista. En 1950 Turing aborda desde otra perspectiva la cuestión: “¿Puede una máquina pensar?”. Turing propone un test (bautizado con su nombre) de caracterización basado en la capacidad de emular el comportamiento humano en el llamado juego de imitación. El test de Turing se define de la siguiente manera:

“Disponemos a un humano y a una máquina en habitaciones diferentes. Un observador les hace una serie de preguntas a uno y a otro a través de la puerta. Si pasado un cierto tiempo el observador no es capaz de determinar quién es el humano y quién la máquina, podemos concluir diciendo que la máquina posee inteligencia.”

Existen defensores y retractores de este test, pero en definitiva es el comúnmente aceptado por la inmensa mayoría de investigadores en IA.

## Definición del campo: conferencia de Dartmouth y los años dorados

En la conferencia de Dartmouth de 1956, una nueva generación de investigadores (Minsky, McCarthy, Newell, Simon, Samuel, Rochester, Shannon, Solomonoff, Selfridge, More) definen las directrices y líneas de actuación futuras, tomando como hipótesis de trabajo la siguiente proposición: *todo aspecto de aprendizaje o cualquier otra característica de inteligencia puede ser definido de forma tan precisa que puede construirse una máquina para simularlo*. En este marco se presenta el Logic Theoristic y, con él, surge el primer área de trabajo, la demostración automática. De esta conferencia se deriva un estado eufórico, propio por otra parte del comienzo de cualquier tecnología novedosa, que propicia aseveraciones desmesuradas y prometedoras en exceso.

En los años siguientes la investigación se centra en la resolución de problemas concretos. Se mejora el Logic Theoristic y se proponen nuevos enfoques cognitivos. En 1958 surgirá LISP, el primer lenguaje de programación propio de IA. Este lenguaje tiene como particularidad que el tipo de datos básico que maneja es la lista. No en vano LISP es el acrónimo de LISt Processing. Un programa en LISP también es una lista, por lo que éste puede utilizar otro (o, incluso, a sí mismo) como entrada para modificarlo o ampliarlo, estando dotado el lenguaje, de esta manera, de la capacidad de aprendizaje.

Los logros de esta etapa se completan con el programa Simbolic Automatic INTEGRATOR (SAINT) de Slage, el primer programa de juegos de Samuel, y los logros en reconocimiento de patrones con el Perceptrón de Rosenblatt y el primer planteamiento de su regla de aprendizaje.

## Las conquistas de los micro-mundos

Durante los sesenta, se avanza en la representación del conocimiento. En esta época se suceden aproximaciones sobre la resolución de problemas concretos como ANALOGY (problemas de analogía), STUDENT, SIR y SINTEX (lenguaje natural), y se produce la formulación de las redes de discriminación de Feigenbaum, las redes semánticas de Quillian y las directrices para el procesamiento semántico de Minsky. No obstante el proyecto que mejor define esta época es el *Blocks Micro-Worlds* del MIT: en un mundo de formas geométricas, los robots interpretan la escena, se mueven, manipulan bloques y explican sus experiencias. En este contexto se desarrolla en 1969 el programa SHRDLU de Winograd, como aportación a la comprensión del lenguaje natural y, un año más tarde, el programa ARCHES de Winston, dedicado al aprendizaje estructural. Otros esfuerzos significativos son: el proyecto SHAKY del SRI, dedicado a la construcción de un robot móvil; y el proyecto de Programación Heurística (HPP) de la Universidad de Stanford, comenzado en 1965 y en cuyo ámbito se desarrollará DENDRAL, el primer sistema experto.

En otro orden de cosas, también se avanza en contextos como la programación de juegos. En 1966, Greenblat comienza a desarrollar un programa para el juego del ajedrez capaz de competir en torneos. En esta época la IA se institucionaliza teniendo lugar en el primer congreso internacional y apareciendo en 1970 el primer número de la revista *Inteligencia Artificial*.

A pesar de los avances sucedidos durante estos años, los investigadores en IA no ven cumplidas, desde el principio, sus expectativas. Ya en 1961, Lucas plantea, desde el ámbito de la Lógica, serias objeciones contra las posibilidades de desarrollo de sistemas inteligentes. Sin embargo, el choque más drástico tiene como protagonista la traducción automática. Efectivamente, desde su planteamiento por Weaver en 1949, los resultados prácticos obtenidos durante los cincuenta y mediados de los sesenta son realmente escasos. En 1966 el informe ALPAC disuade a las instituciones norteamericanas de continuar financiando los proyectos de traducción automática. Este hecho tendrá repercusión en otras investigaciones posteriores de IA, tanto en América como en Europa. Prueba de ello es que a mediados de los setenta se detiene el programa DARPA *Speech Understanding Research*, desarrollado fundamentalmente por la Universidad de Carnegie-Mellon

Adicionalmente, en 1968 Minsky y Papert ponen de manifiesto los límites teóricos de los perceptrones. En 1969, McCarthy y Hayes plantean el problema del marco de referencia y establecen la base del problema de la cualificación. Ese mismo año Weizenbaum desata la polémica al construir un programa aparentemente inteligente: ELIZA. En realidad dicho programa simulaba, utilizando diversos esquemas de lenguaje, un razonamiento inteligente. Más adelante, se descubrirán las limitaciones de generalización de las técnicas aplicadas en micro-mundos. Todas estas críticas motivan una mayor y mejor aproximación de los investigadores a los problemas que aún quedan por resolver.

## Años de crítica y madurez: los difíciles años setenta

El informe ALPAC, los libros de Dreyfus y Weizenbaum, y algunos resultados poco prometedores en el campo hacen que se produzca una recesión y un desinterés casi generalizado en temas de IA.

En este período los investigadores de IA aprenderán de los fracasos y buscarán nuevos enfoques para resolver los viejos problemas. En 1971, Nilsson y Fikes, apoyados por SRI, completan el desarrollo de STRIPS, núcleo planificador del SHAKY. En el período de 1972 a 1976 DARPA financia investigaciones sobre reconocimiento del habla. En 1973 Shank propone una teoría para expresar la

dependencia conceptual y la aplica a la comprensión del lenguaje natural desarrollando el programa MARGIE. En esta época Marr comienza el desarrollo de la primera teoría sobre visión artificial; Duda y Hart publican los primeros textos sobre análisis de escenas; y, en 1975, ARPA financia un programa de comprensión de imagen bajo el que se desarrolla el sistema ACRONYM. En 1975 cristalizan los primeros intentos de definir lenguajes lógicos: Colmenauer define PROLOG y Kowalski incorpora la formulación clausal. En 1976 Lenat escribe su programa de aprendizaje por descubrimiento AM/EURISKO. Por otro lado, los algoritmos para juegos siguen evolucionando.

Esta década conoce también el desarrollo de los sistemas expertos. Al primer sistema, DENDRAL, le seguirán otros como MYCIN o XCON, el primer sistema experto aplicado en ámbitos comerciales. En 1976 Davis aporta el sistema TEIRESIAS, donde se plantea el uso de metaniveles de conocimiento para actualizar bases de conocimiento. Adicionalmente se avanza en aspectos de ingeniería del conocimiento.

En el aspecto de ciencia base, la IA progresó en la definición de teorías de la inteligencia computacional. En 1976, Newell y Simon formulan la *Hipótesis del Sistema de Símbolos Físicos*; Marr publica su enfoque personal; Hayes publica *The Naïve-Physics Manifesto*.

## **Etapa de expansión: los años ochenta**

En esta década se reconoce el trabajo realizado y la IA, por medio de los sistemas expertos y la ingeniería del conocimiento, salta a la industria al mismo tiempo que se convierte en una fuente de iniciativas empresariales. El éxito de XCON cataliza el proceso. Se desarrollan nuevos sistemas como PROSPECTOR, y lenguajes de adquisición e ingeniería del conocimiento como KAS. Poco después, el alto desarrollo del mercado del PC y las exageradas promesas comerciales frenarán el desarrollo de este nuevo sector que se recuperará en pocos años tras volver a planteamientos más racionales. Otros dos factores de índole científico influyen en este receso. Por un lado, se descubren grandes deficiencias en estos sistemas, como su falta de flexibilidad o la dificultad de representar el conocimiento y de diseñar métodos de adquisición. Por otro, el resurgimiento de las redes neuronales tras la mejora del modelo perceptrón, y la aparición de memorias asociativas y de otros modelos motiva la reflexión sobre la adecuación del paradigma simbólico para la resolución de determinados problemas. Las redes neuronales, junto con la lógica difusa, traerán un aire nuevo y su desarrollo conjunto propiciará una nueva generación de sistemas inteligentes y arquitecturas de computación. Por otra parte se suceden los progresos en programación de juegos y se desarrollan nuevas áreas como el estudio de la simulación del sentido común, poniéndose en marcha macro-proyectos como el Cyc, al tiempo que la IA estrecha sus lazos con la teoría cognitiva y se ponen en marcha los proyectos de computadores de 5<sup>a</sup> generación.

## **Estado actual**

Durante los noventa el paradigma conexionista sigue avanzando en compañía de la lógica difusa y los algoritmos genéticos, motivándose el desarrollo de sistemas híbridos, a los que trata de dotar de adaptatividad. Se aportan nuevas metodologías de adquisición del conocimiento, como KADS. En aprendizaje se producen avances significativos y se proponen nuevos métodos. Continúa el desarrollo de la programación declarativa tanto en el desarrollo de métodos automáticos de prototipado como en la propuesta de enfoques de interpretación del lenguaje natural. En cuanto a arquitecturas cognitivas

se concreta la revolución del planteamiento reactivo en el desarrollo de agentes autónomos. Finalmente, en visión artificial se observa un cambio de paradigma, desde el enfoque pasivo clásico al enfoque activo, se conecta la tarea perceptual con la ejecución de acciones (binomio percepción-acción) estudiándose sus implicaciones en el desarrollo de sistemas robóticos con mejores prestaciones.

No obstante, a pesar de los avances conseguidos en los escasos cincuenta años de vida de esta disciplina, los problemas clásicos (percepción, lenguaje natural, juegos, demostración, etc.) siguen siendo objeto de investigación y el desarrollo de una teoría unificada de la inteligencia queda aún lejos.

En general, se ha abandonado el objetivo de buscar una inteligencia artificial, como meta primordial. En vez de eso, se pretende resolver problemas más concretos y que puedan ser fácilmente transferibles a la industria para que, poco a poco, los sistemas vayan adquiriendo inteligencia unido ello al desarrollo tecnológico y al progreso en campos afines (Neurofisiología, Psicología o Biología).

# 2

# Búsqueda heurística

## CONTENIDO DEL CAPÍTULO

- Sección 2.1: *simulated annealing*.
- Sección 2.2: algoritmo  $A^*$ .
- Sección 2.3: juegos:  $\alpha - \beta$  y MTD-f.
- Sección 2.4: satisfacción de restricciones.
- Sección 2.5: ejercicios.
- Sección 2.6: referencias bibliográficas.

## 2.1 *Simulated Annealing*

### Problemas combinatoriales

**IA y NP-completitud** Una de las razones de ser de la IA, si se la compara con la algorítmica más tradicional, es que en ocasiones supone el último reducto para resolver, en un tiempo aceptable, y por ende de forma aproximada, ciertos problemas de gran dificultad. Dichos problemas se denominan *NP-Completos* porque no se conoce algoritmo de coste polinomial, aunque el grado del polinomio sea elevado, que los resuelva. A lo largo de este libro prestaremos atención a algunos de estos problemas ya que aparecen de forma natural en todas las áreas de aplicación de IA. Por citar sólo un par de ejemplos, en *scheduling*, el problema de encontrar la asignación de tareas a múltiples procesadores de forma que se minimice el coste total de ejecución y, en visión artificial, el de encontrar el emparejamiento de mínimo coste entre grafos de objeto durante el reconocimiento, son problemas NP-Completos.

**El viajante de comercio** Uno de estos problemas es el del *viajante de comercio* o TSP<sup>1</sup>. Informalmente, se trata de encontrar el *tour* o ciclo de mínimo recorrido que pasa *por todas* las ciudades de un determinado conjunto visitándolas *una sola vez*. En la Figura 2.1 mostramos nueve capitales de Europa occidental. Así, cuando intentamos evaluar la complejidad del problema nos damos cuenta de que si asignamos un identificador entero a cada ciudad (1=Madrid, 2=París, y así sucesivamente) un ciclo viene determinado por una cadena de identificadores sin repetición cuya posición en la cadena referencia el orden de visita (por ejemplo, 123456789 y 134258976 y 246813597 son tres ciclos diferentes que se cierran volviendo a la ciudad 1). Por lo tanto, para este problema existen tantos ciclos como permutaciones de nueve elementos, esto es,  $9! = 362,880$ , y por ello el TSP tiene un *espacio de configuraciones* de orden factorial. Así, si tenemos 30 ciudades y consumimos  $1\mu\text{s}$  ( $10^{-6}\text{s}$ ) en computar la bondad de cada recorrido, tardaríamos cerca de  $10^{20}$  años en evaluar el espacio de configuraciones completo (según las últimas estimaciones, el Universo *sólo* tiene una edad de  $1,4 \times 10^{10}$  años).

## Relajación estocástica

**Función de energía** Una vez queda claro el espacio de configuraciones, que denotaremos por  $\Omega$ , lo siguiente es formular una función matemática, denominada *función de energía* (en breve veremos por qué) que cuantifique el coste de cada una de sus configuraciones. En el TSP es bastante sencillo encontrar esta función, ya que si tenemos  $N$  ciudades, las distancias entre las ciudades  $c_i$  y  $c_j$  son  $d(c_i, c_j)$ , y cada ciclo  $C \in \Omega$  viene dado por la permutación  $\pi(1)\pi(2)\dots\pi(N)$ , donde  $\pi(i) \in \{1, 2, \dots, N\}$ ,  $\pi(i) \neq \pi(j) \forall i \neq j$ , entonces el coste de dicho ciclo es:

$$E(C) = \sum_{i=1}^{N-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(N)}, c_{\pi(1)}) \quad (2.1)$$

**Distribución de Gibbs** Dada la función de coste  $E(C) \geq 0 : \forall C \in \Omega$ , el algoritmo *Simulated Annealing* asume que la probabilidad de una determinada configuración  $C$  decae exponencialmente con su coste siguiendo la conocida *distribución de Gibbs*:

$$P(C) = \frac{1}{Z} e^{-E(C)/T}, \text{ donde } Z = \sum_{C \in \Omega} e^{-E(C)/T} \quad (2.2)$$

es un factor de normalización denominado *función de partición*, y  $T$  un parámetro de control denominado *temperatura*. Cuando  $T \rightarrow \infty$  resulta que  $P(C)$  es uniforme, de tal manera que todas las configuraciones tienden a ser equiprobables independientemente de su coste. En cambio, cuando  $T \rightarrow 0$  ocurre que  $P(C)$  se concentra en los picos de la distribución de Gibbs, de tal manera que sólo las configuraciones de mínimo tienen probabilidad no nula. En cualquier caso, para una misma temperatura la probabilidad de una determinada configuración decae exponencialmente con su coste y dicho decaimiento es atenuado precisamente por la temperatura.

La relación entre este modelo probabilístico y la búsqueda de la configuración de mínimo coste surge de la analogía entre la resolución de un problema de optimización y la simulación del proceso

---

<sup>1</sup>Traveling Salesperson Problem.



(a)

	(1)MAD	(2)PAR	(3)LON	(4)AMS	(5)COP	(6)BRL	(7)VIE	(8)ROM	(9)BRN
(1)MAD		1,5	2	2,5	5	3	3	2	1
(2)PAR			0,5	0,7	3	2	2,5	2	1
(3)LON				1,2	3	3	3,5	3	2
(4)AMS					1	1	1,5	2,5	2
(5)COP						0,8	1,9	3	2,5
(6)BRL							1	2	1,5
(7)VIE								1	1
(8)ROM									0,8
(9)BRN									

(b)

**Figura 2.1.** TSP entre nueve ciudades europeas. (a) Situación en el mapa en donde representamos el ciclo de coste mínimo 123456789 y otro ciclo 132456789. (b) Tabla de distancias aproximadas, en horas de avión.

de *templado*<sup>2</sup> de una sustancia. Dicha sustancia se expone primero a altas temperaturas y después se enfriá lentamente con objeto de encontrar la configuración de sus partículas con *mínima energía* correspondiente a estructuras reticulares. Sin embargo, la clave del éxito está en el ritmo al que se baja la temperatura: si dicho ritmo es demasiado rápido se alcanzan configuraciones de baja energía, correspondientes a estructuras no regulares sino amorfas y por ello *no* de mínima energía. Estas situaciones equivaldrían, en el mundo de la optimización, a la obtención de mínimos locales en lugar del mínimo global. Por contra, si se elige un buen esquema de descenso de temperatura, el algoritmo *simulated annealing* garantiza convergencia al deseado óptimo global.

<sup>2</sup>Annealing, en inglés.

El algoritmo *simulated annealing* ha de constar de dos elementos clave que desarrollaremos a continuación: (1) Un método que permita obtener nuevas configuraciones a partir de las actuales, y que en definitiva proporcione un esquema de exploración del espacio de configuraciones; y (2) Un esquema de descenso de temperatura que se comporte correctamente, esto es, que garantice la convergencia a óptimos globales.

**Muestreo: algoritmo de Metropolis** Como ya hemos recalcado, el espacio  $\Omega$  es tan gigantesco que huelga un recorrido exhaustivo. En vez de eso se trata de explorarlo selectivamente. Para empezar, supongamos que la temperatura  $T$  es constante. Dada una configuración inicial  $C^0$ , vamos a estar interesados en generar una secuencia de configuraciones  $C^0, C^1, C^2, \dots$ , hasta que demos con la configuración de mínima energía para dicha temperatura. Ya que la configuración de mínima energía es también la de máxima probabilidad, una forma de encontrarla es *tomar muestras*  $C^t$ , con  $t = 1, 2, \dots$ , de  $P(C)$ , de manera que aquellas configuraciones mejores tendrán mayor probabilidad de salir. Cuando el número de muestras es suficientemente elevado, y es de esperar que sea mucho menor que el tamaño de  $\Omega$ , eventualmente generaremos la configuración óptima para  $T$ <sup>3</sup>.

Por el contrario, en lugar de muestrear directamente  $P(C)$  resulta más fácil modelar la secuencia anterior como una *cadena de Markov* y generar  $C^{t+1}$  de acuerdo con la *probabilidad de transición*  $P(C^{t+1}|C^t)$ , eso sí, construyendo dicha probabilidad de forma que se asegure que la cadena converge a la verdadera distribución  $P(C)$ , en este caso la distribución de Gibbs, si el número de muestras es suficientemente elevado. El conocido *algoritmo de Metropolis* aporta una estrategia de muestreo de este tipo y se caracteriza por dos aspectos básicos. En primer lugar, la configuración  $C^{t+1}$  se genera *a partir de*  $C^t$ , de tal forma que  $C^{t+1} \in \mathcal{N}(C^t)$  siendo  $\mathcal{N}(C^t)$  el conjunto de configuraciones vecinas o localmente próximas a  $C^t$  (en un TSP podemos obtener una permutación vecina o local con tan sólo intercambiar las posiciones de dos de los identificadores de las ciudades en el ciclo). Así, el hecho de que configuraciones sucesivas sean similares implica que la exploración de  $\Omega$  puede verse como la realización de un *camino aleatorio* que termina en el óptimo. En segundo lugar, la nueva configuración  $C^{t+1}$  así generada no se asume directamente como buena sino que se acepta *con una cierta probabilidad* definida como:

$$q = \alpha(C^{t+1}|C^t) = \min \left\{ 1, \frac{P(C^{t+1})}{P(C^t)} \right\} \quad (2.3)$$

Siendo  $P(C)$  una distribución de Gibbs, tenemos que para una  $T$  fija se cumple:

$$\frac{P(C^{t+1})}{P(C^t)} = \frac{e^{-E(C^{t+1})/T}}{e^{-E(C^t)/T}} = e^{-\Delta E/T} \quad (2.4)$$

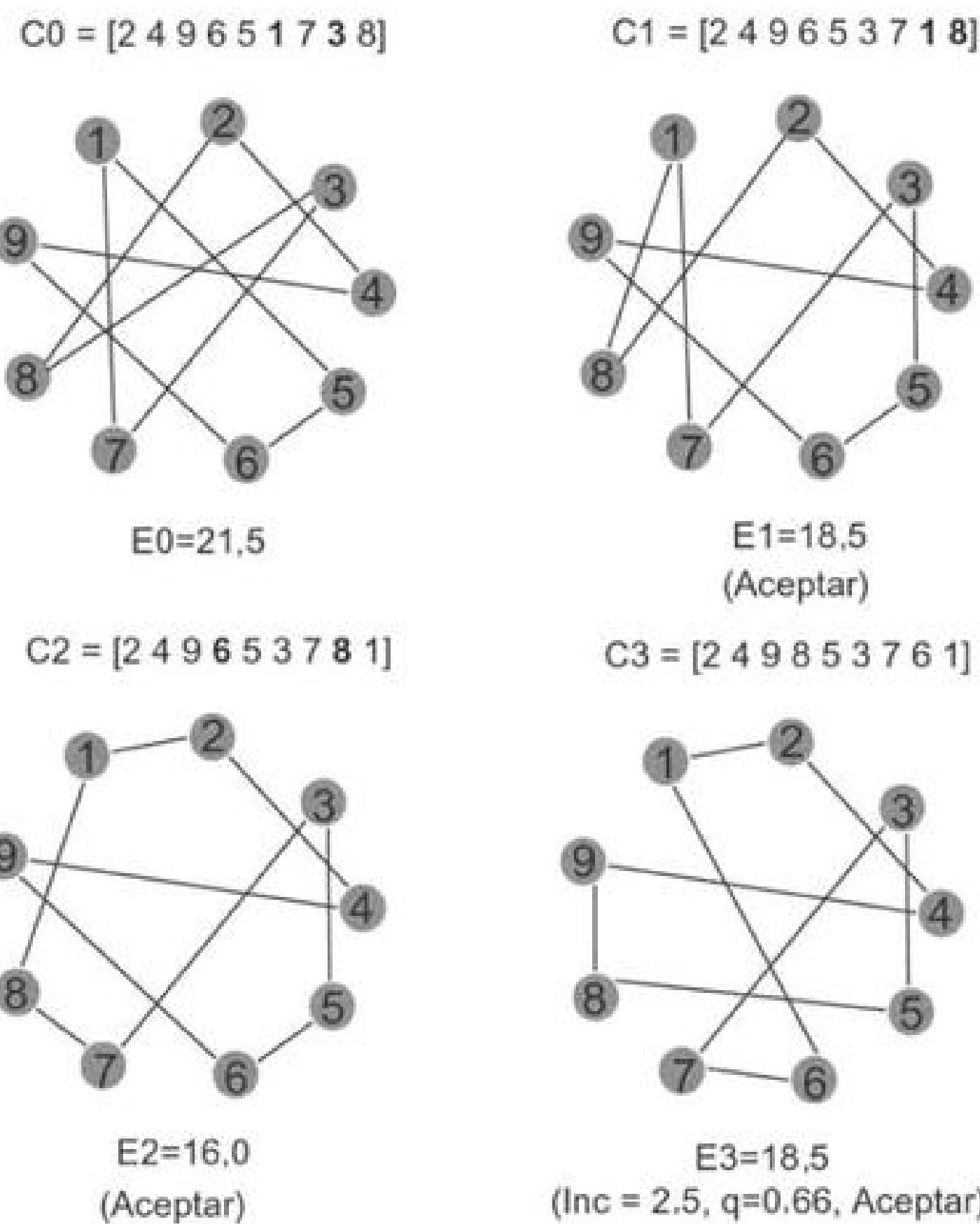
Siendo  $\Delta E = E(C^{t+1}) - E(C^t)$  el incremento de energía debido a la nueva configuración. Si  $\Delta E \leq 0$ , entonces el algoritmo elige sin duda este cambio de configuración ya que indica una caída de energía. Por el contrario, si  $\Delta E > 0$ , entonces la probabilidad de aceptar la nueva configuración decae exponencialmente con la magnitud del empeoramiento relativo  $\Delta E$  y dicho decaimiento está

---

<sup>3</sup>Más adelante, cuando presentemos el enfoque de la computación evolutiva para el aprendizaje, volveremos sobre este tipo de estrategia.

atenuado por  $T$ . En la práctica, la implementación de la regla *aceptar con probabilidad  $q$*  consiste en extraer un número  $r$  de una distribución uniforme definida sobre  $[0, 1]$ , y entonces aceptar la nueva configuración si  $r < q$ .

El funcionamiento de este algoritmo para el problema del TSP se ilustra en la Figura 2.2. Partiendo de la configuración  $C^0 = [2\ 4\ 9\ 6\ 5\ 1\ 7\ 3\ 8]$ , que tiene una energía  $E^0 = 21,5$ , se obtiene la  $C^1 = [2\ 4\ 9\ 6\ 5\ 3\ 7\ 1\ 8]$  permutando las posiciones ocupadas por el 3 y el 1 respectivamente. Esto da lugar a una mejora de energía, puesto que  $E^1 = 18,5$ . Consecuentemente,  $C^1$  se acepta puesto que  $\alpha(C^1/C^0) = 1$ . Lo mismo ocurre con  $C^2 = [2\ 4\ 9\ 6\ 5\ 3\ 7\ 8\ 1]$ , que se obtiene de  $C^1$  permutando el 8 y el 1, dando lugar a una energía  $E^2 = 16,0$ . Sin embargo, la configuración  $C^3 = [2\ 4\ 9\ 8\ 5\ 3\ 7\ 6\ 1]$ , que se obtiene a partir de  $C^2$  intercambiando el 6 y el 8, tiene una energía  $E^3 = 18,5$ , con lo que  $\Delta E = 2,0$ , lo cual representa un incremento significativo. Si  $T = 6,0$ , encontramos que  $q = \alpha(C^3/C^2) = e^{-2,5/6,0} = 0,66$ , con lo que las probabilidades de aceptar esta configuración son superiores a la mitad. Sin embargo, si la temperatura desciende a la mitad, esto es,  $T = 3,0$ , entonces resulta que  $q = \alpha(C^3/C^2) = e^{-2,5/3,0} = 0,43$  y, si  $T = 1,0$  tendríamos que  $q = 0,08$ .



**Figura 2.2.** Ilustración del algoritmo Metropolis. Las dos primeras iteraciones llevan a una mejora de energía mientras que la siguiente lleva a un empeoramiento, pero esta configuración finalmente es aceptada.

**Algoritmo *simulated annealing*** El algoritmo de Metropolis genera una secuencia de configuraciones  $C^0, C^1, \dots$  y se detiene cuando se satisface una condición de equilibrio que en la práctica se traduce en alcanzar un número determinado de muestras. La última muestra generada por el algoritmo puede tomarse como una muestra de la distribución de Gibbs para una temperatura  $T$  dada. Para que dicha muestra sea el mínimo global de la función de energía es preciso realizar una sucesión de muestreos de Metropolis de forma que la temperatura  $T$  vaya decreciendo monótonamente de un muestreo al siguiente, hasta que se satisfaga  $T \approx 0$ . Dicha sucesión de muestreos es lo que se denomina como algoritmo *simulated annealing*, cuyo pseudocódigo se muestra en la Figura 2.3. En él hacemos referencia a las funciones INICIALIZAR\_CONFIG e INICIALIZAR\_TEMP, que proporcionan la configuración inicial (normalmente elegida de forma aleatoria) y la temperatura inicial, respectivamente. La función GENERAR proporciona una permutación local de la configuración que se le pasa como argumento. La función RANDOM devuelve un valor uniformemente distribuido en el rango que se le pasa como argumento. El control de la convergencia de Metropolis lo realiza la función booleana TEST\_EQUIBALRIO. Finalmente, ENFRIAR disminuye la temperatura de acuerdo con un esquema que garantice la convergencia al óptimo global.

---

```

Algoritmo SIMULATED_ANNEALING {
     $C^0 \leftarrow \text{INICIALIZAR\_CONFIG};$ 
     $T \leftarrow \text{INICIALIZAR\_TEMP};$ 
     $t \leftarrow 0;$ 
    Repetir {
         $C \leftarrow C^t;$ 
        Repetir {
             $C' \leftarrow \text{GENERAR}(\mathcal{N}(C));$ 
             $\Delta E \leftarrow E(C') - E(C);$ 
             $q \leftarrow \min\{1, e^{-\Delta E/T}\};$ 
            Si ( $\text{RANDOM}(0,1) < q$ ) {  $C \leftarrow C'$  };
        } Hasta TEST_EQUIBALRIO;
         $t \leftarrow t + 1;$ 
         $C^t \leftarrow C;$ 
         $T \leftarrow \text{ENFRIAR}(T, t);$ 
    } Hasta  $T \approx 0;$ 
    Devolver  $C^t \approx C^* = \arg \min_{C \in \Omega} E(C);$ 
}

```

---

**Figura 2.3.** Algoritmo *simulated annealing* como anidamiento de muestreos de Metropolis.

Sin duda la característica más interesante de este algoritmo es su *capacidad de evitar mínimos locales*, lo cual se debe a la cuidadosa gestión del parámetro  $T$ . La clave está en el cálculo de la probabilidad de aceptación  $q = \min\{1, e^{-\Delta E/T}\}$  y en la decisión a tomar cuando  $\Delta E > 0$ , es decir, cuando estamos ante un posible mínimo local. Como el decaimiento exponencial de dicha probabilidad está atenuado por  $T$ , cuando  $T \rightarrow \infty$  o es suficientemente elevada (lo cual suele ocurrir en las

primeras iteraciones del algoritmo), la probabilidad de aceptación es alta por lo tanto podemos escapar del mínimo local. Sin embargo, como la temperatura decrece a medida que el algoritmo progresá, también decrece la atenuación del decaimiento exponencial y, así, para un mismo  $\Delta E > 0$  resulta cada vez más difícil escapar del mínimo local (de lo contrario podríamos escapar del mínimo global) hasta que esto es totalmente imposible cuando  $T \rightarrow 0$ , ya que entonces se entiende que el algoritmo se encuentra en las proximidades del mínimo global. Así pues, en las primeras fases del algoritmo, éste tiene un comportamiento bastante aleatorio y a medida que progresa el descenso de la temperatura se transforma en un método voraz.

Veamos el funcionamiento de este algoritmo en nuestro problema TSP. Teniendo en cuenta que el número de iteraciones hasta declarar estabilidad en Metropolis es de 100, y que el máximo de iteraciones de *annealing*, en donde  $T \approx 0$ , es de 10.000, en la parte superior de la Figura 2.4 se muestran los valores de la función de energía para todas las configuraciones que han sido visitadas por el algoritmo. Para poder visualizar la función de energía en tres dimensiones hemos representado solamente los valores de las dos primeras posiciones de la configuración, esto es,  $\pi(1)$  y  $\pi(2)$ , es decir, hemos reducido un espacio de 9 dimensiones a 2 dimensiones. A pesar de la imprecisión que supone representar 123456789 por el mismo punto en dicho espacio que 124356789 obtenemos una función bastante regular no exenta de mínimos locales y en la que destaca un mínimo global de coste de 9,3.

Por otro lado, en la parte inferior de la Figura 2.4 mostramos la  $q = \alpha C^{t+1}/C^t$ , promedio a lo largo de las 100 iteraciones de cada Metropolis. Como puede verse claramente, la probabilidad de aceptación desciende paulatinamente a medida que aumenta el número de iteraciones, y con ello la temperatura.

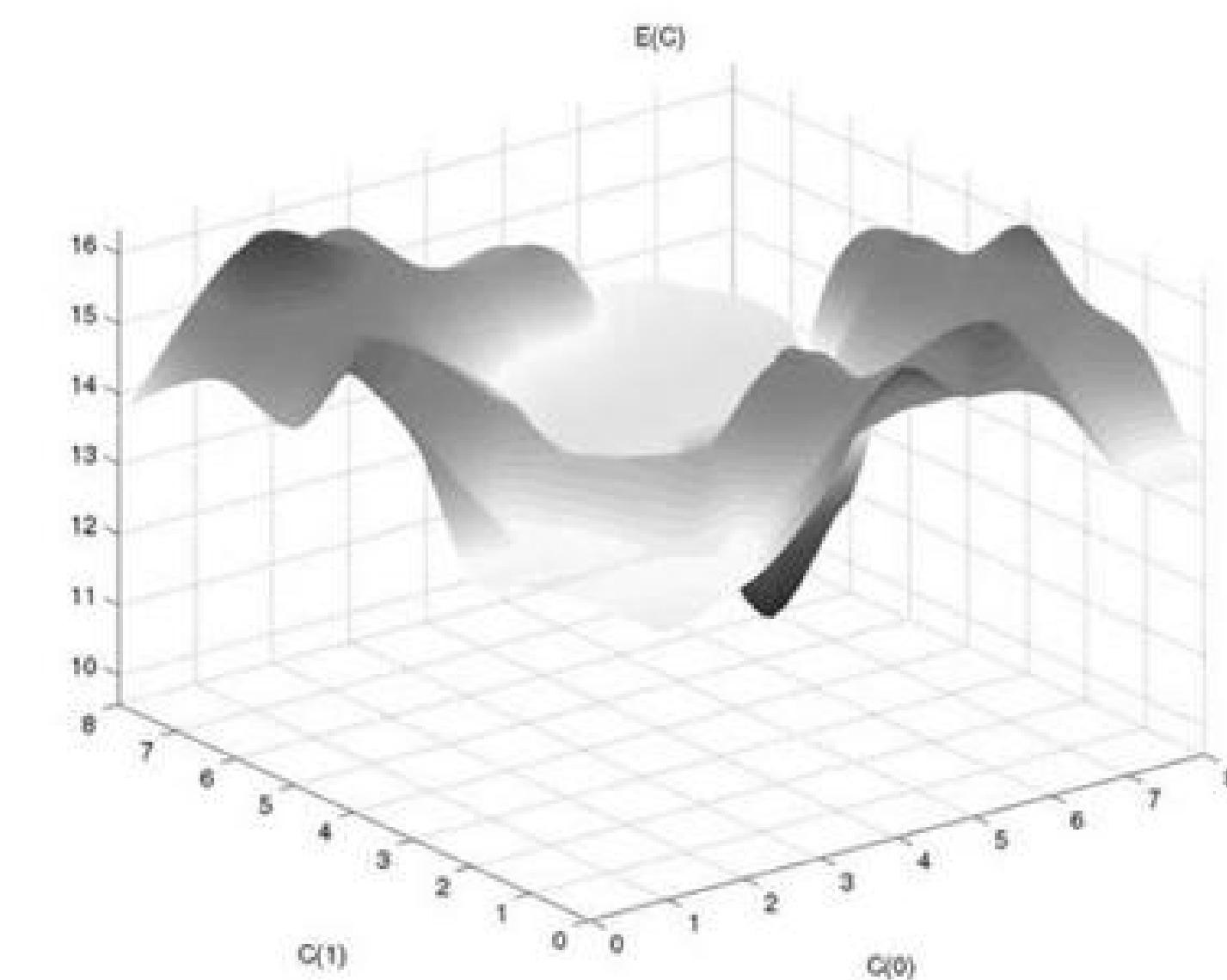
**Convergencia: comportamiento asintótico** Como hemos visto, el adecuado funcionamiento del algoritmo depende de la elección del esquema de descenso de temperatura, es decir, de la implementación de la función ENFRIAR. Por un lado, si se elige un esquema que satisfaga las siguientes condiciones:

$$\lim_{t \rightarrow \infty} T^{(t)} = 0 \text{ y } T^{(t)} \geq \frac{N \times \Delta}{\ln(1+t)} \quad (2.5)$$

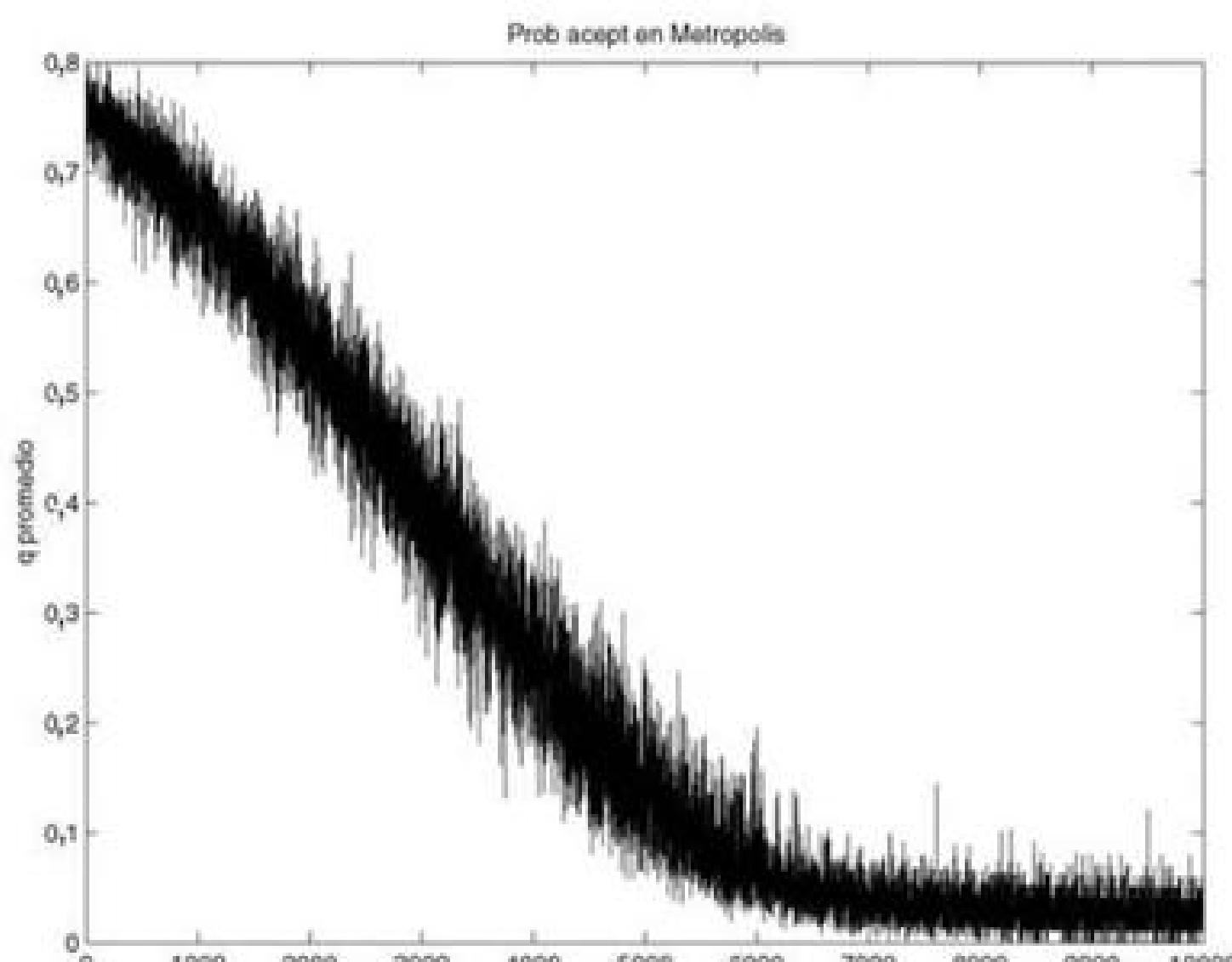
donde  $N$  es el número de elementos de una permutación del problema (número de ciudades en el TSP) y  $\Delta = \max_{C \in \Omega} E(C) - \min_{C \in \Omega} E(C)$ , entonces se garantiza la convergencia al mínimo global independientemente de la configuración inicial,  $C^0$ , elegida (algo así como *todos los caminos conducen a Roma*). Sin embargo, este esquema resulta demasiado lento, incluso para un pequeño espacio de configuraciones. Por ejemplo, para  $N = 5$ , y  $\Delta = 1$ , llevaría  $e^{10} \approx 22,027$  muestreos de Metropolis para alcanzar  $T = 0,5$ , lo cual supone explorar un 6 % del espacio total. Estos números son dramáticos a poco que aumente  $N$  ya que, para  $N = 9$  se deberían emplear  $e^{18} \approx 65,659,969$  muestreos, lo cual supera con creces la dimensión del espacio de búsqueda.

Alternativamente, si se sustituye el producto  $N \times \Delta$  por una constante  $C$  y se sustituye la desigualdad por igualdad tenemos el esquema:

$$T^{(t)} = \frac{C}{\ln(1+t)}, \text{ con } t = 0, 1, 2, \dots \quad (2.6)$$



(a)



(b)

**Figura 2.4.** Funcionamiento del *annealing*. (a) Función de energía asumiendo las dos primeras componentes de cada configuración por donde pasa la búsqueda. (b) Probabilidad de aceptación promedio del algoritmo Metropolis para cada temperatura.

No obstante, la constante  $C$  debe ajustarse experimentalmente. Por ejemplo, en ciertos problemas de análisis de imagen, esta constante puede tomar valores como 3,0 y 4,0. Éste es por tanto un criterio de naturaleza *heurística*. En IA, una heurística es, en sentido amplio, cualquier artificio capaz de

limitar el espacio de búsqueda. Esta denominación incluye frecuentemente que se pierda la garantía de alcanzar el resultado óptimo en aras de obtener una solución en un tiempo aceptable<sup>4</sup>.

Otra posibilidad, también heurística, es el siguiente esquema, todavía más rápido que el anterior (se elimina la inversa del logaritmo):

$$T^{(t)} = \kappa T^{(t-1)} \quad (2.7)$$

donde  $\kappa$  es una constante que usualmente toma valores entre 0,8 y 0,99, y la temperatura inicial  $T^0$  se elige de tal forma que todos los cambios de configuración propuestos se acepten. Siguiendo esta línea tenemos, finalmente, otro esquema:

$$T^{(t)} = T^0 \left( \frac{T^f}{T^0} \right)^{\frac{t}{t_{\max}}} \quad (2.8)$$

en donde  $T^0$  y  $T^f$  son las temperaturas inicial y final, respectivamente, y  $t_{\max}$  denota el número máximo de muestreos que se desea realizar. Este esquema se denomina *geométrico* en contraposición con el logarítmico clásico.

En la Figura 2.5 comparamos el funcionamiento de los esquemas logarítmico y geométrico en la resolución del TSP mediante el *annealing*. Teniendo en cuenta que la temperatura nula se alcanza a las 10.000 iteraciones, en la parte superior de la figura se muestra la convergencia del esquema geométrico para dicho número de iteraciones. En este caso se alcanza el óptimo, cuyo coste es de 9,3. Como puede apreciarse en la parte inferior de la figura, cuando se utiliza el esquema logarítmico con ese número de iteraciones no se logra convergencia. Sin embargo, conviene tener en cuenta que el esquema geométrico tiene, en general, una garantía de convergencia menor que el logarítmico, aunque en la práctica, y especialmente en problemas tan complejos como éste, sea más preferible que el logarítmico.

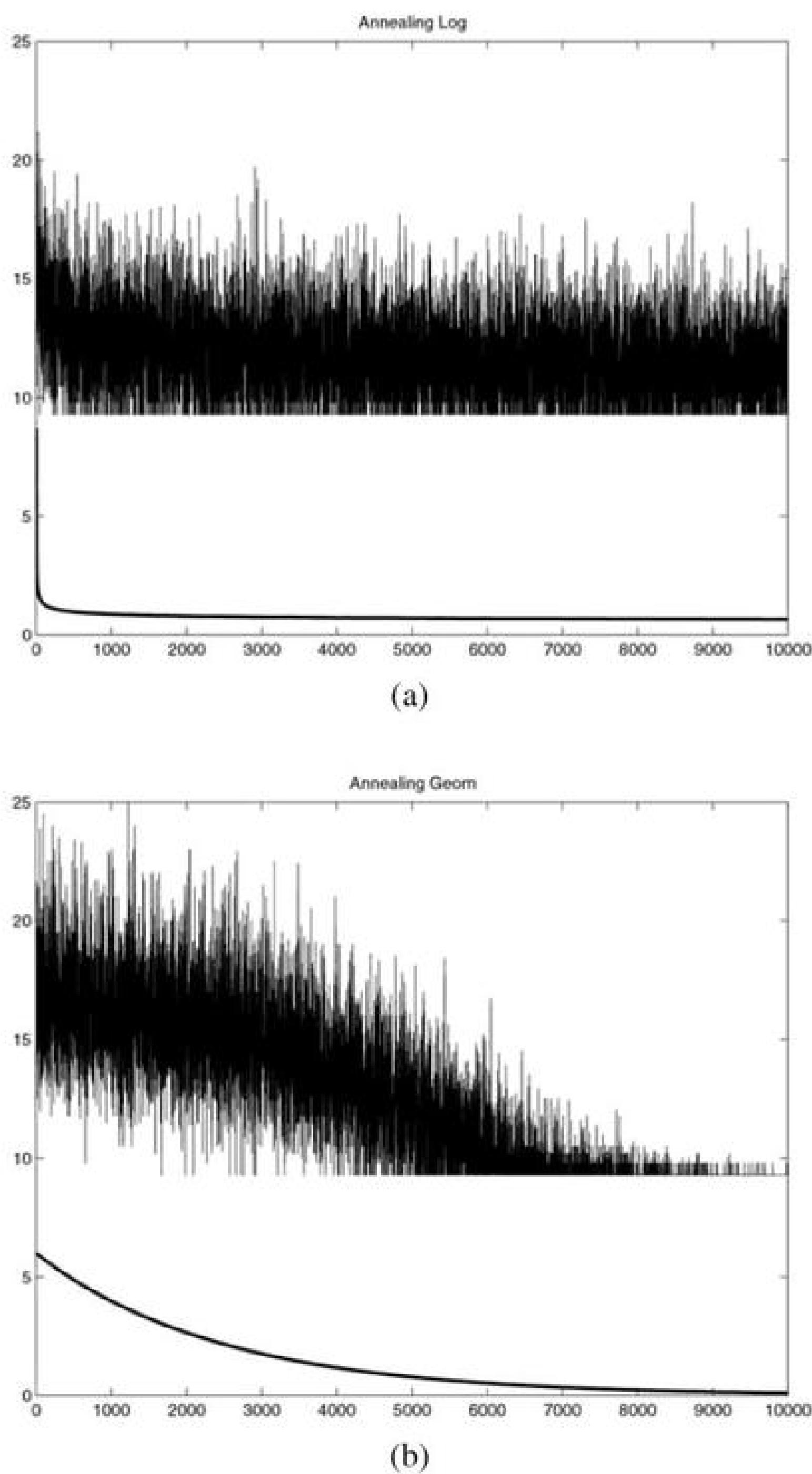
## 2.2 Algoritmo A\*

### Concepto de heurística

**Pathfinding en juegos** *Pathfinding*, esto es, la búsqueda o planificación de caminos, se ha convertido en un elemento esencial en la programación de juegos de computador. Un ejemplo sencillo es el clásico juego de *Pacman*. Un buen programa de computador debería encontrar la trayectoria más corta desde cada fantasma a la posición del *come-cocos* y planificar el movimiento consecuentemente en lugar de moverse aleatoriamente hasta dar con él. Otro ejemplo más reciente es el juego de estrategia *Age of Empires* (véase la Figura 2.6). El objetivo del juego es conquistar el mundo con una de las civilizaciones disponibles. Para ello se parte de la situación caótica resultante de la caída del imperio romano y se intenta construir un nuevo imperio, lo cual implica explorar y potencialmente enfrentarse con otras civilizaciones a través de diversas campañas. En estas campañas la planificación de caminos se utiliza para mover unidades de combate por el terreno evitando obstáculos, etc. La planificación de caminos es una forma exhaustiva de analizar el terreno de combate. Las deficiencias en este tipo

---

<sup>4</sup>Más adelante distinguiremos entre heurísticas admisibles y no-admisibles para diferenciar precisamente los casos en los que se garantiza solución óptima de aquéllos en los que esta garantía se pierde.



**Figura 2.5.** Comparación de esquemas de temperatura para TSP. (a) Evolución de la función de energía para el esquema logarítmico. (b) Evolución para el caso geométrico. (En ambos casos la temperatura asociada a cada iteración se muestra bajo la gráfica de energía.)

de lógica de la primera versión (de hecho no era capaz de salvar obstáculos para llegar al enemigo) fueron muy criticadas y se corrigieron en *Age of Empires II*. No obstante, en la nueva versión se invierte cerca del 70 % del esfuerzo de simulación en planificación de caminos.



**Figura 2.6.** Pantalla del juego de computador *Age of Empires*.

**Funciones de evaluación** Tomemos como ejemplo el problema del análisis del terreno en juegos como *Age of Empires*. De manera general tenemos que encontrar el *camino de coste mínimo* entre la posición origen y la posición destino a través del terreno. El coste del camino puede venir dado por distintos criterios. Si, como puede verse en la parte superior de la Figura 2.7, asociamos coste con altura sobre el terreno, y tomamos el convenio de que puntos con elevada intensidad representan poca altura y viceversa, entonces la identificación del camino más corto exigirá el trazado de diversos caminos hacia la solución y su evaluación (marcamos en blanco los puntos por los que ha pasado alguno de estos caminos). Así, los obstáculos, como ríos, árboles, etc., se corresponderán (ver parte inferior de la figura) con bloques de puntos de color negro<sup>5</sup>.

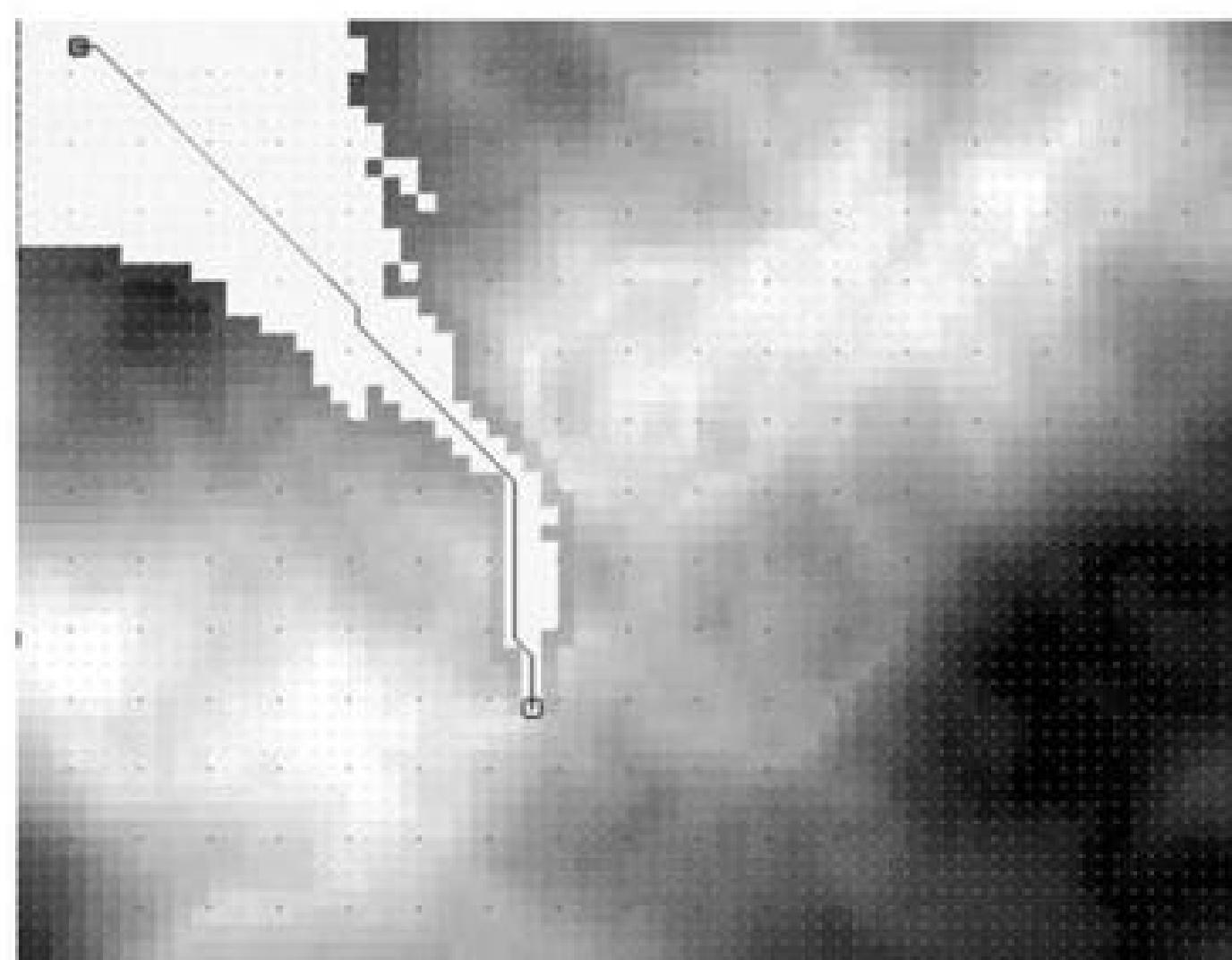
No obstante, la estrategia de búsqueda que se ha empleado en estos ejemplos no se corresponde con *definir la función de energía en términos de la suma de costes y muestrear el espacio de caminos posibles* como ocurre con el *simulated annealing*. Alternativamente, en lugar de evaluar un camino completo en cada iteración, podemos mantener una población de caminos *parcialmente desarrollados* que partan del punto inicial  $I$  y extender preferentemente los caminos parcialmente desarrollados más prometedores hasta que uno de ellos alcance la posición final  $F$ . Denotando por  $n$  los nodos o posiciones del mapa que representan el final momentáneo de caminos parcialmente desarrollados, elegiremos como nodo más prometedor en cada iteración aquel que minimice el valor de una determinada *función de evaluación*  $f(n)$ . Siguiendo la formulación aditiva clásica de dicha función, ésta es la suma de dos términos. El primero evalúa la distancia recorrida desde  $I$  hasta  $n$ , mientras que el segundo considera la distancia que queda por recorrer desde  $n$  hasta  $F$ . Así,  $f(n)$  viene dada por:

$$f(n) = g(n) + h(n) \quad (2.9)$$

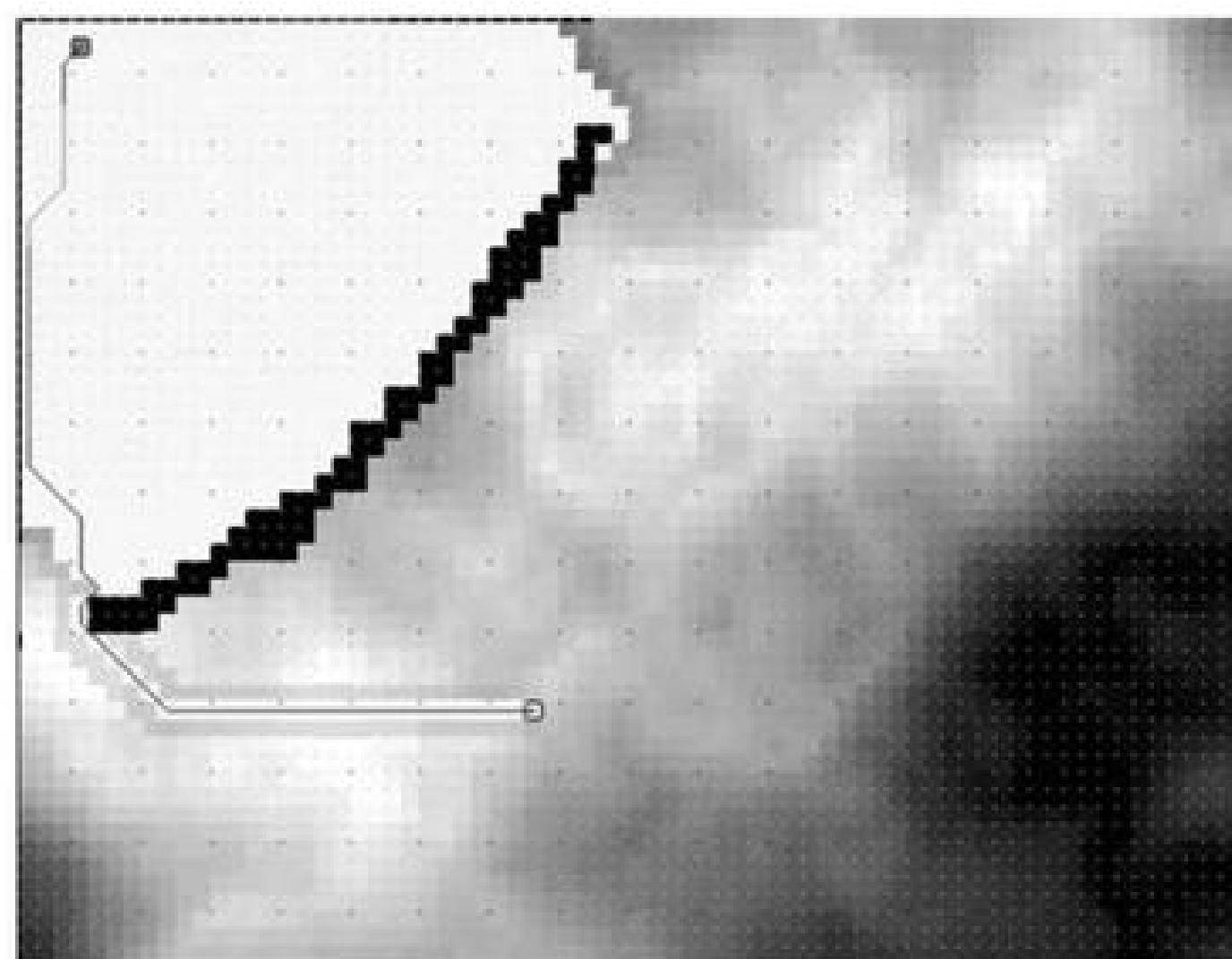
<sup>5</sup>Para estas simulaciones se ha utilizado el programa *A\* Explorer* de James Matthews, que puede descargarse de <http://www.generation5.org>.

En primer lugar,  $g(n)$  es *una estimación* de  $g^*(n)$ , coste del camino más corto desde  $I$  hasta  $n$ . Por ejemplo, en nuestro problema  $g(n)$  podría ser la suma de los costes de las casillas pertenecientes al camino que momentáneamente termina en  $n$ . No obstante, puede haber otros caminos parciales que terminen en  $n$ , y que aún no hemos explorado, cuya suma de longitudes sea incluso menor. Por lo tanto,  $g(n) \geq g^*(n)$ , es decir que  $g(n)$  es una estimación pesimista, pero es la menos pesimista que podemos realizar hasta el momento.

A su vez,  $h(n)$  es la *heurística*, esto es, la estimación de  $h^*(n)$ , coste del camino más corto desde  $n$  hasta  $F$ . Al contrario de lo que ocurre con  $g(n)$ , encontrar una buena formulación para  $h(n)$  puede resultar una ardua tarea, ya que desconocemos la estructura de los caminos que parten de  $n$  y alcanzan  $F$



(a)



(b)

**Figura 2.7.** Análisis del terreno con *pathfinding*. (a) Búsqueda del camino de mínimo peso sin obstáculos. (b) Búsqueda con obstáculos.

y por lo tanto desconocemos la distancia que todavía hemos de recorrer hasta alcanzar  $F$ . Una función heurística sencilla de calcular sería  $\|n - F\|$ , la distancia en línea recta entre  $n$  y  $F$ . En los ejemplos de análisis del terreno que se muestran en la Figura 2.7, se ha utilizado esta distancia como  $h(n)$  y la suma de costes acumulados como  $g(n)$ . El funcionamiento de esta heurística es razonablemente adecuado ya que en situaciones en las que varios caminos parciales tienen un valor de  $g(n)$  idéntico se prioriza la extensión o expansión de los caminos cuyo último nodo está más cerca de  $F$ . Ésta es la esencia del concepto de heurística: *proporcionar una regla general que anticipa el mejor camino a seguir y que permita por tanto descartar caminos no-óptimos también lo antes posible*. Esta regla será tanto mejor cuanto antes se produzca dicha anticipación. Así,  $h^*(n)$  representa la forma óptima de realizar dicha anticipación y por tanto el esfuerzo de búsqueda realizado en términos del número de nodos explorados cuando se utiliza esta heurística es el mínimo esfuerzo requerido para encontrar el camino óptimo en este esquema y equivale a disponer de un informador perfecto y, por tanto, a una situación ideal. Sin embargo no es oro todo lo que reluce. En primer lugar, es muy difícil encontrar una formulación matemática para el informador perfecto. En segundo lugar, el esfuerzo de búsqueda de realizar una evaluación de  $h^*(n)$  suele ser elevadísimo (la información cuesta y disponer de un informador perfecto no siempre compensa.)

## Algoritmo $A^*$ sobre grafos

**Estructuras auxiliares** Antes de proseguir con las implicaciones formales derivadas del tipo de función de evaluación que acabamos de ver, vamos a detallar el funcionamiento del algoritmo de búsqueda heurística  $A^*$  sobre grafos. Así pues, y teniendo en cuenta que en un grafo podemos llegar a un mismo nodo a través de distintos caminos, pero solamente nos interesa registrar el camino de llegada de mínimo coste hasta el momento, mantendremos un *árbol de búsqueda* denominado *Arbol*. Dicho árbol mantendrá un conjunto de punteros que garanticen un único camino desde  $I$  hasta cada final de camino parcial  $n$  que se considere para su expansión. Dicho camino será el de mínimo coste descubierto hasta el momento. Por otro lado, la estructura en árbol nos garantizará la ausencia de ciclos y por tanto de la posibilidad de re-visitar algún nodo.

Además, el procedimiento de búsqueda mantendrá dos estructuras de datos más. Por un lado, está la lista *Frontera* que contiene los nodos  $n$  finales de caminos parcialmente desarrollados que se considerarán en cada iteración para su continuación o expansión, es decir, aquéllos sobre los que se aplicará la función de evaluación  $f(n)$ . Esta lista representa el frente de avance de la búsqueda. Por otro lado, tenemos la lista *Interior* que contiene los nodos interiores. Así, cuando se decida extender un camino parcial  $n$  porque resulta más prometedor que los demás se incluirán en la frontera todos los nodos accesibles desde  $n$  que no hayan sido visitados.

**Algoritmo  $A^*$**  El algoritmo (Figura 2.8) comienza inicializando el árbol de búsqueda con el nodo inicial  $n_0$  (en nuestro ejemplo la posición inicial del robot  $I$ ). Dicho nodo, el primero que se va a considerar, también se incluye en *Frontera*. A continuación se inicializa la lista *Interior* como vacía. A partir de este punto se entra en un ciclo de búsqueda que irá desarrollando los caminos más prometedores, uno por cada pasada. La primera tarea en cada pasada consiste en testar si la frontera está vacía, en cuyo caso se produce una salida reportando error, ya que ello significa que no existen caminos por desarrollar. A continuación se extrae el primer nodo de *Frontera* (EXTRAER\_PRIMERO) y se inserta al final de la lista *Interior* (INSERTAR\_FINAL). Éste siempre será el que represente el camino

---

```

Algoritmo  $A^*$  {
     $Arbol \leftarrow n_0;$ 
     $Frontera \leftarrow n_0;$ 
     $Interior \leftarrow \emptyset;$ 
     $Exito \leftarrow False;$ 
    Repetir {
        Si ( $Frontera = \emptyset$ ) { DEVOLVER_ERROR };
         $n \leftarrow EXTRAER_PRIMERO(Frontera);$ 
         $Interior \leftarrow INSERTAR_FINAL(Interior, n);$ 
        Si ( $n = n_f$ ) {  $Exito \leftarrow True$  };
        Sino {
             $Sucesores \leftarrow EXPANDIR(n);$ 
             $Arbol \leftarrow INCORPORAR_SUCESES(Arbol, Sucesores);$ 
             $Arbol \leftarrow REORGANIZAR_PUNTEROS(Sucesores);$ 
             $Frontera \leftarrow EVALUAR_Y_REORDENAR(Frontera, f());$ 
        }
    } Hasta  $Exito$ ;
Devolver  $C_{min}$ ;

```

---

**Figura 2.8.** Algoritmo de búsqueda  $A^*$  en grafos.

parcialmente desarrollado más prometedor, ya que mantenemos la frontera ordenada de acuerdo con este criterio. A este nodo lo llamamos  $n$ . Pues bien, si resulta que  $n$  es el nodo final  $n_f$  (en nuestro ejemplo el que representa la posición  $F$ ) entonces hemos terminado y ya podemos salir del bucle. En caso contrario realizamos las siguientes actualizaciones:

- Obtener con **EXPANDIR** la lista *Sucesores* que contiene los nodos descendientes de  $n$  (accesibles desde dicho nodo), que no son ascendientes de ningún nodo de *Arbol*.
- Incorporar los elementos de *Sucesores* como descendientes de  $n$  en *Arbol* (**INCORPORAR\_SUCESES**).
- Para aquellos elementos de *Sucesores* que no pertenezcan ni a *Frontera* ni a *Interior* se procederá a crear un puntero hacia  $n$ . Este puntero indica que el camino más corto desde  $n_0$  pasa por  $n$ . Así, aquellos elementos de *Sucesores* que pertenezcan a *Frontera* o *Interior* ya apuntan hacia algún nodo y solamente procederemos a cambiar dichos punteros hacia  $n$  en el caso de que la ruta de llegada a ellos a través de  $n$  tenga menor coste que la ruta a través del nodo al que actualmente apuntan. Esta reorganización de punteros debe extenderse a los descendientes en *Arbol* de todos los elementos de *Sucesores* que ya estaban en *Interior*. Todo ello se realiza en **REORGANIZAR\_PUNTEROS**.
- Aplicar la función de evaluación y reorganizar la lista *Frontera* de manera que esté ordenada de menor a mayor valor de  $f(n)$  (**EVALUAR\_Y\_REORDENAR**).

Si el algoritmo termina con éxito, acaba devolviendo los punteros desde  $n_f$  hasta  $n_0$  y dichos punteros son los que forman el camino de coste mínimo  $C_{min}$ , lo cual demostraremos en dos etapas. En primer lugar demostraremos que el algoritmo termina incluso para grafos infinitos y, a continuación, demostraremos que encuentra  $C_{min}$ , para lo cual bastará con que la heurística  $h(n)$  cumpla una cierta condición.

**Ilustración del algoritmo** Mostraremos el funcionamiento del algoritmo  $A^*$  con un ejemplo sencillo de análisis del terreno (véase la situación inicial descrita en la Figura 2.9). Supongamos un terreno donde cada casilla tiene coste unidad (en blanco)<sup>6</sup>. Los obstáculos (en negro) marcan celdillas *no explorables*. En todo momento el nodo seleccionado para expandir se indica con una flecha. Asumimos  $h(n) = ||n - F||$ , es decir, la distancia euclídea entre el nodo actual y el nodo final  $F$ <sup>7</sup>. Pues bien, en la primera iteración se selecciona para expandir el nodo  $I$  (el único elemento de *Frontera*) y una vez expandido se coloca en *Interior*. La expansión de  $I$  consiste en sus 8 celdas vecinas. Gráficamente, en todo momento las celdas/nodos que pertenecen a *Frontera* se caracterizan porque existe algún vecino no visitado (en caso contrario, estas celdas forman parte de *Interior*).

Tras expandir  $I$  e incorporar sus sucesores a *Frontera* se produce la evaluación. El nodo  $n$  de menor  $f(n) = g(n) + h(n)$  se indica con una flecha. De todos los vecinos de  $I$  que tienen la misma  $g(n)$  se elige aquel que más cerca está de  $F$ , y por tanto de menor  $h(n)$ . En las iteraciones 2 – 5 no ocurre nada especialmente reseñable salvo que va avanzando paulatinamente hacia la solución. No obstante, para garantizar que el camino óptimo no se pierde<sup>8</sup> se observa que se produce un desarrollo equilibrado del árbol de búsqueda: si bien en la iteración 4 cruzamos el obstáculo y avanzamos hacia la solución [lo que significa que  $g()$  aumenta y  $h()$  disminuye] en la iteración siguiente expandimos un nodo de menor profundidad [menor  $g()$  y mayor  $h()$ ] en lugar de seguir profundizando.

Por otro lado, en la Figura 2.10, podemos observar cómo se construye la estructura *Arbol*. Hasta la iteración 7 no se produce ningún replanteamiento de los punteros de algún nodo. El nodo etiquetado como  $A$  en la iteración 6 va a cambiar su puntero en la iteración siguiente. Lo mismo ocurre con el nodo etiquetado como  $B$  en la iteración 7, que cambia su puntero en la iteración 8. Por otro lado, en la iteración 8 vemos cómo se selecciona para expandir un nodo que está muy cerca de  $F$ . No obstante  $F$  no se descubre hasta que es seleccionado para expandir en la iteración siguiente. Dado que  $A^*$  debe garantizar optimalidad, podría darse el caso de seleccionar antes para expandir algún nodo de menor profundidad. No obstante, en este ejemplo se selecciona para expandir el nodo  $F$  (no lo hemos indicado con una flecha) y posteriormente se devuelve el camino trazado por los punteros.

## Admisibilidad de $A^*$

**Terminación** Asumiendo que cada nodo tiene un número finito de sucesores y que el coste de cada arista del grafo es mayor que  $\delta > 0$ , el algoritmo que acabamos de esbozar no sólo *termina* para grafos finitos (la propia construcción de una estructura acíclica *Arbol* garantiza que eventualmente seleccionaremos  $n_f$ ) sino que también lo hace para grafos infinitos. Esto último se debe a que  $g(n)$  aumentará de forma desmedida y eventualmente seleccionaremos el nodo final  $n_f$ .

<sup>6</sup>Para esta traza y las siguientes en esta sección hemos utilizado la herramienta *PathDemo*, desarrollada por Brian Stout y disponible en <http://www.gamasutra.com>.

<sup>7</sup>Cabe utilizar otras distancias como veremos más adelante en la parte de ejercicios.

<sup>8</sup>En la sección siguiente justificaremos formalmente esta garantía.

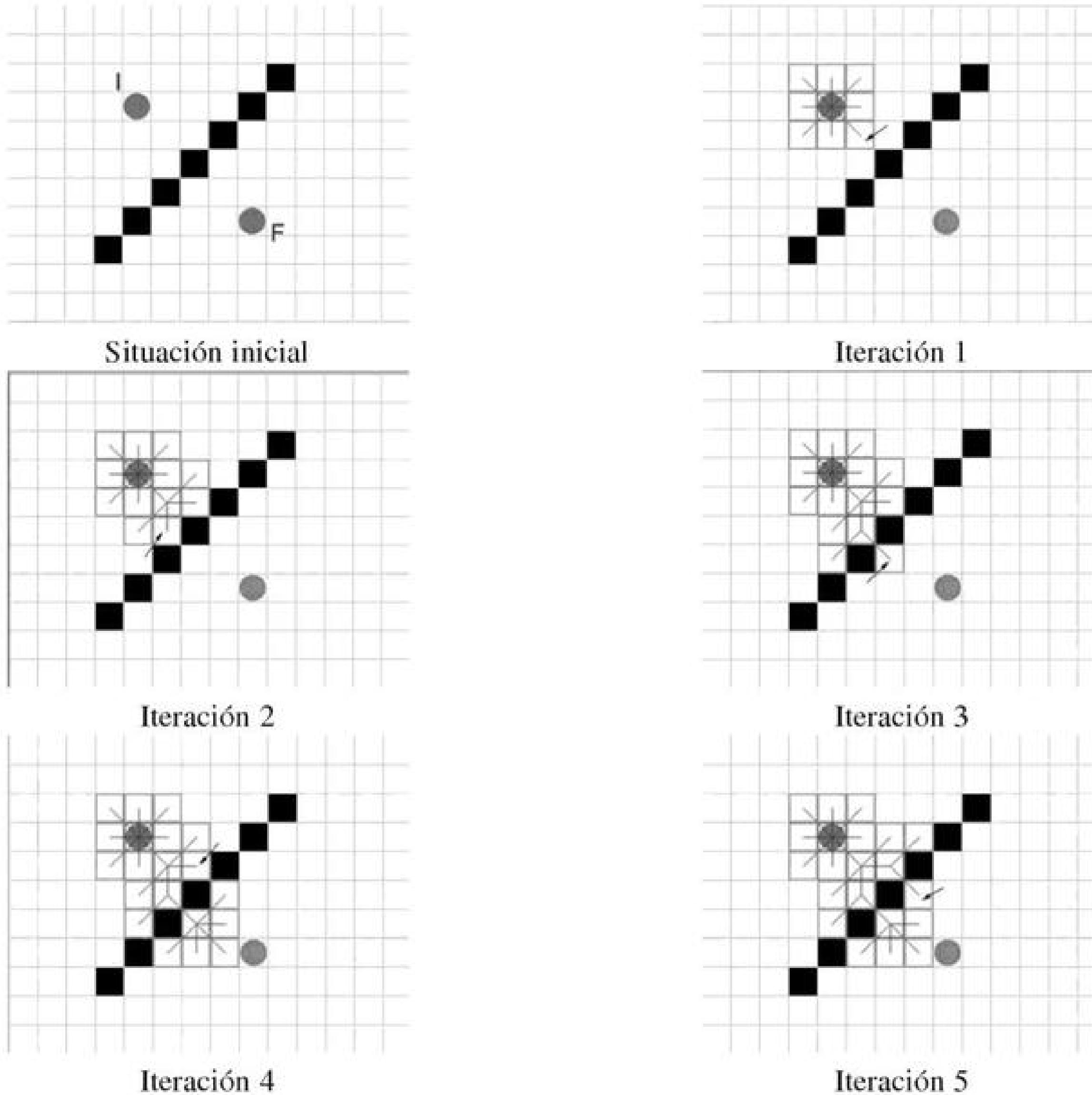
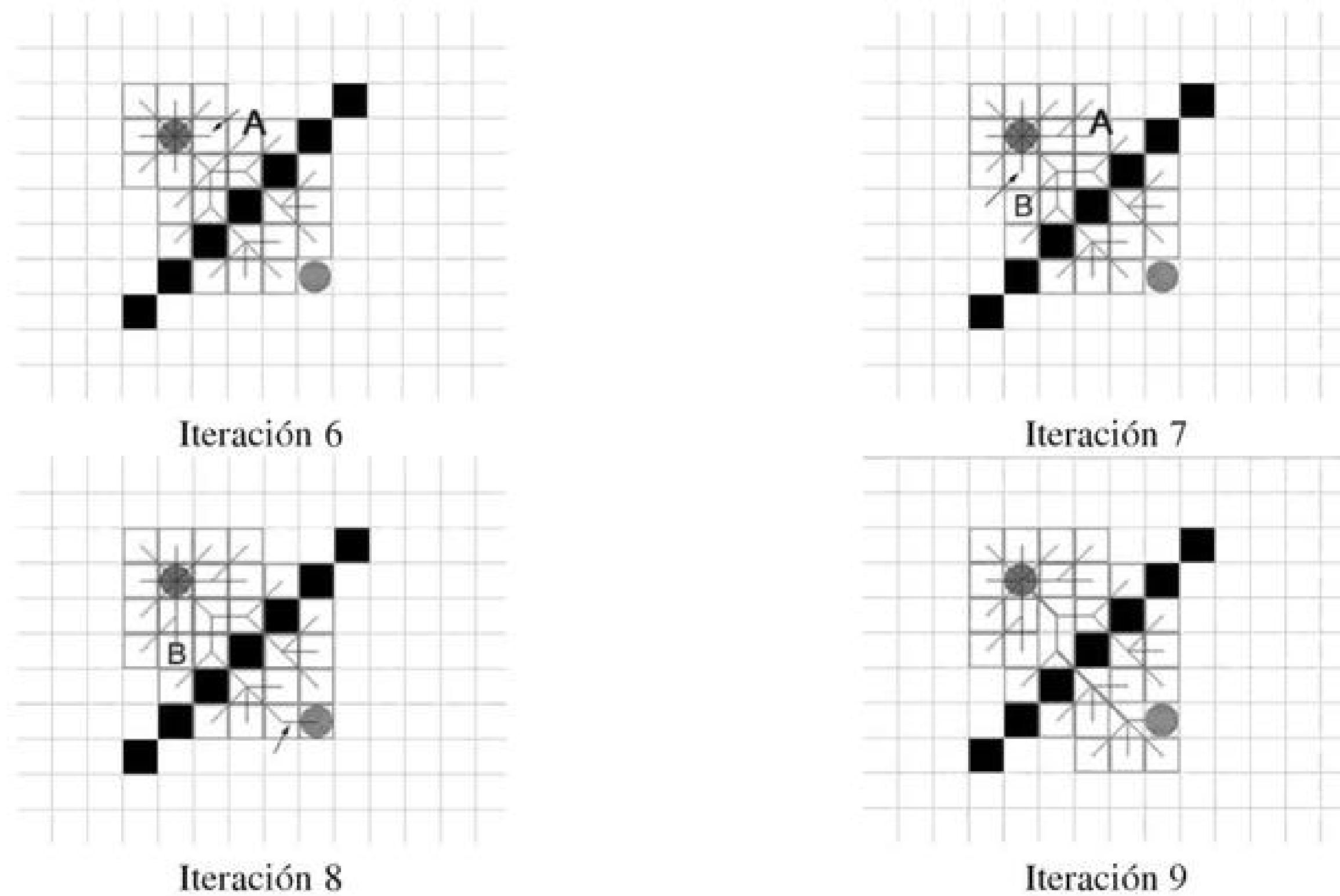


Figura 2.9. Traza del algoritmo  $A^*$ . Iteraciones 1 a 5.

**Optimalidad** La demostración de que  $A^*$  encuentra el camino de coste mínimo  $C_{min}$  es algo más elaborada. En primer lugar supongamos que  $h(n) \leq h^*(n) \forall n$ , esto es, que  $h(n)$  siempre infra-estima  $h^*(n)$  o como mucho da un valor igual a  $h^*(n)$ . Cuando ocurre esto se dice que la heurística es *admissible*. Pues bien, en estas condiciones se puede asegurar que el algoritmo  $A^*$  encuentra  $C_{min}$ . Para probarlo, demostraremos en primer lugar que en cualquier momento antes de la terminación existe en *Frontera* un nodo  $m \in C_{min}$ , tal que  $f(m) \leq C^*$ , donde  $C^*$  es el coste de  $C_{min}$ . En primer lugar, en todo momento hay un nodo  $m \in C_{min}$  en *Frontera* porque el algoritmo no deja ningún camino por explorar parcialmente. Además, como  $m$  está en el camino óptimo, se cumple que  $g(m) = g^*(m)$  y, entonces, haciendo uso de que  $h(n) \leq h^*(n)$  tenemos que:

$$f(m) = g(m) + h(m) = g^*(m) + h(m) \leq g^*(m) + h^*(m) = f^*(m) = C^* \quad (2.10)$$



**Figura 2.10.** Traza del algoritmo  $A^*$ . Iteraciones 6 a 9.

La segunda parte de la demostración se basa en un esquema de reducción al absurdo. Así pues, supongamos que el algoritmo termina devolviendo un nodo  $t$  de  $Frontera$  tal que  $f(t) > C^*$ , el cual representa un camino cuyo coste es mayor que el de  $C_{min}$ . Pues bien, cuando  $t$  fue seleccionado para expandir se cumplía que  $f(t) \leq f(n) \forall n \in Frontera$ , es decir, que era el mejor de todos los que estaban en la frontera (en nuestra implementación, el primer nodo de dicha lista). Por lo tanto si dicho nodo cumple que  $f(t) > C^*$  y tiene un coste menor que todos los de  $Frontera$ , entonces *todos* los nodos  $n \in Frontera$  satisfacen que  $f(n) > C^*$ , lo cual es falso puesto que, como hemos probado anteriormente, al menos uno de ellos llamado  $m$  satisface que  $f(m) \leq C^*$ . Por lo tanto se demuestra que el algoritmo no puede terminar sin descubrir el camino óptimo. Se dice entonces que el algoritmo es *admissible*.

**Nivel de información** En nuestro ejemplo, la heurística  $h(n) = ||n - F||$  de distancia en línea recta podría considerarse admisible, ya que la distancia mínima que queda por recorrer siempre sería superior a  $h(n)$  habiendo obstáculos que bordear (ambas distancias serían iguales en el caso de que existiera una arista en el grafo de  $n$  a  $F$ ). Ya dijimos que esta heurística es útil porque puede evaluarse rápidamente. Si además es admisible, ello la hace interesante por partida doble. Sin embargo, una cuestión interesante es si sería deseable encontrar otra heurística  $h'(n)$ , tal que  $h(n) \leq h'(n) \leq h^*(n) \forall n$ , esto es, una función más cercana a  $h^*(n)$ , es decir, con mayor *nivel de información*. Desde el punto de vista del número de nodos a expandir para encontrar  $C_{min}$  puede demostrarse que el algoritmo con

$h(n)$  expande al menos tantos nodos como con  $h'(n)$ . Se dice entonces que  $h'(n)$  tiene mayor *potencia heurística* que  $h^*(n)$ .

Supongamos que en nuestro problema de análisis del terreno tenemos:

$$\begin{aligned} h'(n) &= ||n - F|| = \sqrt{d_x^2 + d_y^2} \\ h''(n) &= \max(|d_x|, |d_y|) \end{aligned}$$

donde  $d_x$  y  $d_y$  son respectivamente las diferencias entre las coordenadas  $x$  y entre las coordenadas  $y$  de  $n$  y  $F$ . Fácilmente se puede ver que  $h'(n) \geq h''(n) \forall n$ , con lo que la distancia euclídea está más informada que el máximo de los valores absolutos. ¿Qué implicaciones tiene esto a nivel de *coste computacional* de la búsqueda, es decir, de número de nodos expandidos y de coste de evaluación de  $f(n)$ ? Normalmente, una heurística más potente tiene un coste de evaluación mayor (aunque esa diferencia no es apreciable en este caso) pero, en compensación, permite expandir menos nodos. En la parte superior de la Figura 2.11 mostramos la convergencia del algoritmo para  $h'(n)$  (izquierda) y  $h''(n)$  (derecha). Como puede verse, el número de nodos explorados es mayor en el primer caso que en el segundo.

Otro aspecto interesante a tener en cuenta es que, teóricamente, podemos obtener una heurística mejor informada, por ejemplo que  $h'(n)$ , simplemente multiplicando la heurística original por una

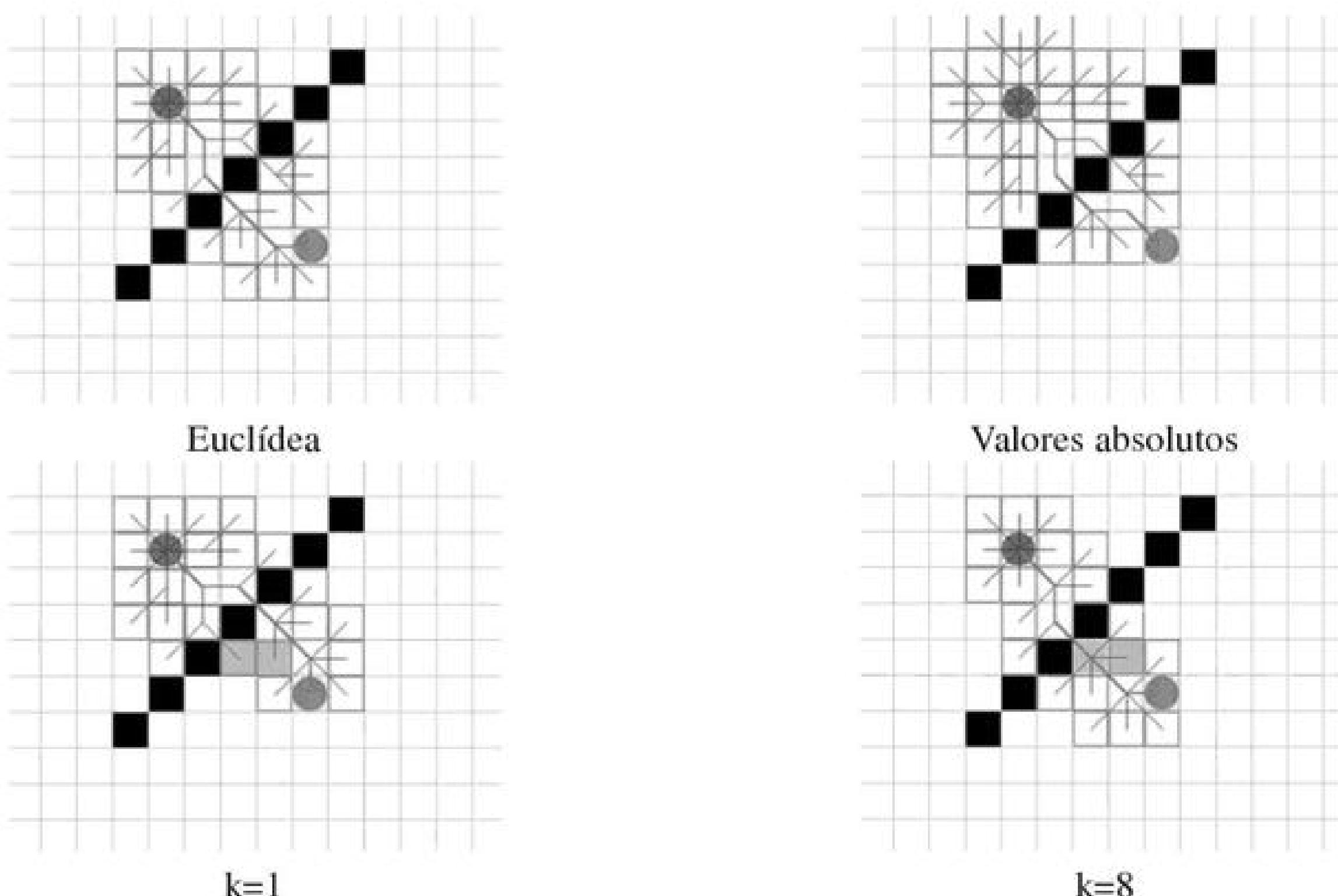


Figura 2.11. Potencia heurística y admisibilidad de  $A^*$ .

constante  $k > 0$ , es decir haciendo que  $h''(n) = kh'(n)$ . No obstante, cuanto mayor sea  $k$  más probable será la admisibilidad de que  $h''(n)$  se pierda a pesar de que  $h'(n)$  sea admisible. En la parte inferior de la Figura 2.11 hemos diseñado un nuevo escenario en donde aparecen dos celdillas en gris, lo que codifica un coste 3 para ellas, en lugar del coste 1 de las celdillas en blanco. Pues bien, resulta que cuando  $k = 1$ , se obtiene el camino óptimo, *que no pasa por ninguna de estas celdillas*, pero cuando  $k = 3$ , el camino obtenido no es el óptimo y pasa por una de esas celdillas. ¿A qué se debe este resultado? Pues a que el *sobrepeso* de  $3h'(n)$  fuerza el algoritmo a considerar que el camino que pasa por la celdilla indicada por la flecha (que pertenece al camino óptimo) es *peor* de lo que realmente es y a que prefiera desarrollar el camino parcial que termina en la primera celdilla gris. Consecuentemente el resultado es un camino no-óptimo.

## Variantes de $A^*$

**Algoritmo sub-óptimo  $A_\varepsilon^*$**  Un grave problema de  $A^*$  es que independientemente de la heurística utilizada, y debido al compromiso con la optimalidad de la solución a obtener (admisibilidad), se mantiene simultáneamente un elevado número de caminos parciales no-óptimos antes de que el algoritmo descubra el camino de coste mínimo. Así, surge la cuestión de si no sería conveniente relajar de alguna forma el compromiso de optimalidad en aras de descubrir antes dicho camino. Sin duda ello conllevaría obtener un coste de  $C'_{\min}$  ligeramente mayor que  $C_{\min}$  (o no tan *ligeramente* mayor dependiendo del nivel de relajación adoptado). Así, diremos que un algoritmo es  $\varepsilon$ -admisible siempre y cuando termine produciendo una solución cuyo coste no exceda del coste mínimo por un factor de  $1 + \varepsilon$ , esto es, cuando:

$$C'_{\min} \leq (1 + \varepsilon)C_{\min} \quad (2.11)$$

con  $\varepsilon \geq 0$ . El algoritmo  $A_\varepsilon^*$  surge de incorporar este planteamiento en  $A^*$ . El planteamiento es muy sencillo. Se trata de mantener otra lista denominada *Focal* que es una sublistas de *Frontera*. Dicha lista *Focal* contiene aquellos nodos  $n$  de la frontera de búsqueda cuya  $f(n)$  no excede la evaluación del mejor nodo de la frontera  $n_{\text{mejor}}$  por un factor de  $1 + \varepsilon$ , esto es:

$$Focal = \{n \in Frontera : f(n) \leq (1 + \varepsilon)f(n_{\text{mejor}})\} \quad (2.12)$$

donde  $n_{\text{mejor}} = \arg \min_{m \in Frontera} \{f(m)\}$ . La justificación de mantener esta lista adicional es el hecho de que todos los nodos que contiene se consideran igualmente prometedores a efectos de continuar la búsqueda (ya que su diferencia en coste es muy escasa). A efectos de este algoritmo, tanto da seleccionar  $n_{\text{mejor}}$  como cualquier otro de la lista *Focal*. Por lo tanto, en lugar de esperar a decidir cuál de ellos es mejor en un futuro, se selecciona para expansión en aquel nodo de *Focal* que minimice una segunda heurística  $h_f(n)$  denominada *heurística focal*. Dicha segunda heurística no precisa ser admisible y de hecho puede contener cualquier estimación del coste necesario para completar la solución. El hecho de que  $h_f(n) > h^*(n)$  no va a comprometer en absoluto la admisibilidad- $\varepsilon$  del algoritmo. De hecho, el paso de la admisibilidad a la admisibilidad- $\varepsilon$  y por tanto la garantía de obtener el óptimo<sup>9</sup> se produce desde el mismo instante en que decidimos que todos los nodos de la lista *Focal*, incluido  $n_{\text{mejor}}$ , son igualmente prometedores antes de utilizar la heurística focal que sirve de hecho para resolver el desempate.

<sup>9</sup>Eso no quiere decir que ocasionalmente no demos con el óptimo.

---

```

Algoritmo IDA* {
    Corte  $\leftarrow f(n_0)$ ;
    Repetir {
         $(Corte, Sol) \leftarrow \text{BUSCAR\_EN\_PROFUNDIDAD}(Corte, n_0)$ ;
        Si ( $Sol \neq \emptyset$ ) { Devolver  $Sol$  };
        Si ( $Corte = \infty$ ) { DEVOLVER_FALLO };
    }
}

Función BUSCAR_EN_PROFUNDIDAD( $Corte, n$ ) {
     $f_{sig} \leftarrow \infty$ ;
    Si ( $Corte < f(n)$ ) { Devolver  $(f(n), \emptyset)$  };
    Si ( $n = n_f$ ) { Devolver  $(Corte, n)$  };
    Para cada  $s \in \text{SUCESORES}(n)$  Hacer{
         $(f_{nueva}, Sol) \leftarrow \text{BUSCAR\_EN\_PROFUNDIDAD}(s, Corte)$ ;
        Si ( $Sol \neq \emptyset$ ) { Devolver  $(Corte, Sol)$  };
         $f_{sig} \leftarrow \min\{f_{sig}, f_{nueva}\}$ ;
    }
    Devolver  $(\emptyset, f_{sig})$ ;
}

```

---

**Figura 2.12.** Algoritmo de búsqueda  $IDA^*$ .

Por tanto, el parámetro  $\epsilon$  nos va a permitir regular niveles de avance en profundidad hacia la solución. Cuando  $\epsilon \rightarrow \infty$  o es suficientemente grande se tiende a pensar que todos los nodos de *Frontera* son igualmente prometedores a falta de aplicar la heurística focal, lo que puede causar que el nodo que conduce al camino óptimo se quede sistemáticamente sin desarrollar. En consecuencia, las desviaciones de  $C_{min}$  serán elevadas. Por contra, cuando  $\epsilon \rightarrow 0$  o es suficientemente bajo, estas desviaciones se reducen hasta que en el límite con  $\epsilon = 0$  tenemos el algoritmo  $A^*$  y la desviación del coste óptimo es también nula.

**Algoritmo  $IDA^*$**  Otro aspecto crítico de  $A^*$  es su excesivo coste espacial. El hecho de mantener un amplio conjunto de caminos parcialmente desarrollados en la estructura *Arbol* conlleva un elevado consumo de memoria. Si tenemos en cuenta que el número de hojas (en este caso nodos de *Frontera*) de un árbol con un número fijo de descendientes  $B$  y profundidad  $l$  es de  $B^l$ , mientras que el número de nodos interiores, de la lista *Interior*, es  $B^0 + B^1 + \dots + B^{l-1}$ , nos haremos una idea clara de que el coste espacial crece exponencialmente con  $l$ . Este hecho limita fuertemente, por ejemplo, el desarrollo de estrategias de planificación de trayectorias en mapas grandes y eso provoca que  $A^*$  consuma casi un 70% de la simulación en *Age of Empires*.

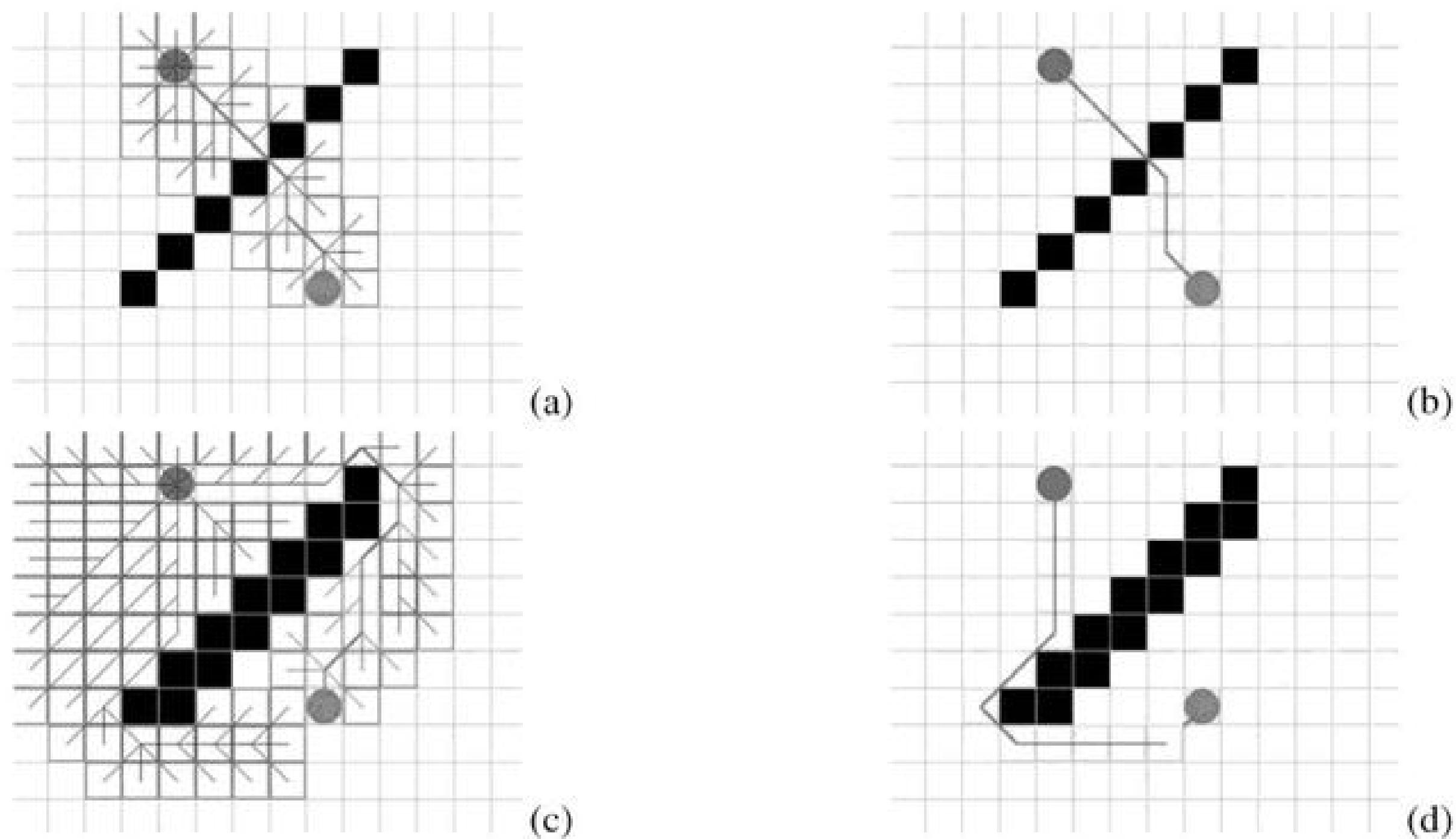
El algoritmo de descenso iterativo  $IDA^{*10}$  tiene por objeto garantizar un coste espacial lineal con la profundidad de la búsqueda. El algoritmo (véase la Figura 2.12) procede definiendo un coste inicial

---

<sup>10</sup>Iterative Deepening  $A^*$ .

denominado *coste de corte* o *Corte* que es igual a  $f(n_0) = g(n_0) + h(n_0) = 0 + h(n_0) = h(n_0)$ , siendo  $n_0$  el nodo raíz o inicial. Si  $h(n)$  es admisible tenemos garantizado que en todo momento  $f(n) \leq f^*(n)$ . Por lo tanto inicialmente se cumplirá que  $Corte \leq C_{min}$  y en consecuencia el algoritmo deberá progresar incrementando paulatinamente *Corte* hasta alcanzar  $C_{min}$ . Así, en cada etapa el algoritmo explorará recursivamente los descendientes de  $n_0$ , tarea que se realizará en **BUSCAR\_EN\_PROFUNDIDAD**. En cada rama la búsqueda en profundidad se frenará cuando alcancemos un nodo  $n$  tal que  $Corte < f(n)$ , es decir, hasta que se alcance un valor interesante para incrementar *Corte*, o bien cuando alcancemos un nodo objetivo  $n_f$ , en cuyo caso se devolverá el *Corte* actual. Cuando todas las ramas que parten de  $n_0$  hayan terminado se tomará como nuevo coste de corte el mínimo de los costes de corte obtenidos para todas las ramas, y entonces comenzará una nueva etapa del algoritmo. Si resulta que en algún momento *Corte* no puede incrementarse más y alcanzamos un nodo solución, entonces es que hemos encontrado el camino mínimo. Dicho camino estará contenido en *Sol*.

En la Figura 2.13 comparamos el esfuerzo de búsqueda y los requerimientos de memoria de  $A^*$  y  $IDA^*$  en dos escenarios de análisis del terreno. En el primer escenario, ambos algoritmos pueden atravesar el obstáculo. No obstante,  $IDA^*$  avanza rápidamente hacia la solución a través de un descenso de profundidad  $d = 7$  y encuentra el camino mínimo mucho más pronto que  $A^*$  con un consumo mínimo de memoria. En la segunda situación, en donde ambos algoritmos deben descubrir uno de los extremos del obstáculo para poder sortearlos, el esfuerzo de búsqueda  $A^*$  y  $IDA^*$  es más parejo.  $A^*$  avanza hacia  $F$  explorando las dos alternativas de camino óptimo, ambas del mismo coste, hasta



**Figura 2.13.** Comparación entre  $IDA^*$  y  $A^*$ . Escenario fácil en donde  $IDA^*$  (b) encuentra mucho más pronto el camino óptimo que  $A^*$  (a). Escenario con obstáculos en donde  $A^*$  (c) e  $IDA^*$  (d) requieren un esfuerzo de búsqueda similar.

que finalmente se decide por el camino que sorteá la parte superior del obstáculo. En cambio, *IDA*<sup>\*</sup> comienza a lanzar búsquedas en profundidad que van *rebotando* en las paredes del obstáculo. El algoritmo debe extender la profundidad para poder subir el *Corte*. Entonces descubre que lo consigue sorteando la parte inferior del obstáculo. Técnicamente, el algoritmo funciona como si de un pulpo con tentáculos gigantes se tratara. En términos de consumo de memoria el algoritmo mantiene simultáneamente menos caminos que *A\**, pero obtiene un resultado exitoso a costa de explorar múltiples rutas, comportamiento que se agudiza a medida que separamos *F* del obstáculo.

## 2.3 Juegos: $\alpha - \beta$ y MTD-f

### Búsquedas MINIMAX y $\alpha - \beta$

**Árboles de búsqueda en juegos** Uno de los pocos dominios de aplicación en donde la IA ha cubierto las expectativas iniciales ha sido sin duda en el desarrollo de algoritmos de búsqueda para el juego del ajedrez<sup>11</sup> a pesar de las considerables dimensiones del espacio de configuraciones para este problema. En efecto, partiendo de la situación inicial las blancas (llamémosles MAX) pueden realizar  $B$  movimientos. A su vez las negras (llamadas MIN) pueden responder a cada uno de estos movimientos con otros  $B$  movimientos, y así sucesivamente hasta que termine la partida, digamos tras realizar cada jugador  $l$  movimientos. Estos movimientos se pueden registrar en un árbol, denominado árbol MINIMAX, que representa la partida completa con  $2 \times l$  niveles alternando un nivel en el que mueven las blancas con otro en el que mueven negras (totalizando  $1 + B + B^2 + \dots + B^{2 \times l}$  nodos) hasta llegar a las  $B^{2 \times l}$  hojas que representan situaciones en las que gana MAX (etiquetadas con  $+\infty$ ) o en las que gana MIN negras (etiquetadas con  $-\infty$ ). En una partida real, inicialmente  $B = 20$  (véase la Figura 2.14a), pero el valor promedio de  $B$  es 35 y el de  $l$  es 50. Entonces, para el problema del ajedrez estamos hablando de explorar alrededor de  $35^{100}$  estados o configuraciones, lo cual resulta impracticable.

**Procedimiento MINIMAX** En vez de desarrollar el árbol MINIMAX de una vez, los programas de ajedrez parten de la situación actual en la que corresponde el turno a un jugador, por ejemplo a MAX, y generan el árbol correspondiente a un *horizonte de juego* de  $k$  niveles. Al llegar a las hojas  $h$  de dicho árbol se dispone de una función de evaluación  $e(h)$  que devuelve valores altos en situaciones en que MAX tiene ventaja y valores bajos en situaciones en que la ventaja es para MIN. Pensemos, por ejemplo, en  $\sum_i V(b_i) - V(n_i)$ , donde  $V()$  es la función que devuelve el valor de una pieza (como 1 para el peón, 5 para el alfil, e  $\infty$  para el rey),  $b_i$  son las piezas blancas y  $n_i$  son las negras. Pues bien, el objetivo de MAX es tomar la decisión (cuyo resultado es uno de los posibles nodos del nivel que sigue inmediatamente a la raíz) que conduce a *maximizar*  $e(r)$ , siendo  $r$  el nodo raíz. Para tomar dicha decisión, se asume que cada jugador juega de forma óptima, que ambos jugadores conocen las reglas del juego, que no existe el azar (esto último permite que el juego sea planificable con total certeza hasta el horizonte de juego elegido), y que el resultado del juego solamente puede ser GANAR, PERDER o EMPATAR. Por ejemplo, en la Figura 2.14b mostramos cómo el computador, jugando con las negras, detecta el *mate del pastor* y desarrolla el peón de dama. Esta situación se ha detectado con un horizonte 2; para detectarla con anterioridad se requeriría un horizonte mayor.

<sup>11</sup> La derrota de Kasparov por Deep-Blue en 1997 es un claro ejemplo de ello.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

que llega al nodo MAX padre de dicho nodo MIN. Esta situación da lugar a lo que se llama un *corte*  $\beta$ . La situación simétrica, cuando el sub-árbol se encabeza por un nodo MAX, se denomina corte  $\alpha$ . Pues bien, cuando cada nodo tiene a su derecha un valor peor, el número de cortes totales es máximo, en consecuencia, también lo es el ahorro de evaluaciones de nodos hoja, de tal manera que el número de evaluaciones se reduce aproximadamente a  $B^{l/2}$ , lo cual quiere decir que podemos descender al doble de profundidad que con minimax en el mismo tiempo. Por el contrario, cuando cada nodo tiene a su derecha un valor peor (véase la Figura 2.16b) no se produce ningún corte y tenemos exactamente el comportamiento del algoritmo MINIMAX.

**Algoritmo  $\alpha - \beta$**  El algoritmo que implementa las podas descritas anteriormente se basa en ajustar dinámicamente (véase la Figura 2.17) los llamados *límite- $\alpha$*  y *límite- $\beta$*  de cada nodo. Para el nodo raíz, dichos límites valen respectivamente  $-\infty$  y  $+\infty$ , ya que con toda seguridad el valor MINIMAX del nodo raíz estará comprendido entre estos límites. Dichos valores se propagan recursivamente hacia abajo y se actualizan. Los nodos MAX actualizan su límite  $\alpha$  tan pronto como reciben un valor  $v > \alpha$  y los nodos MIN actualizan su límite  $\beta$  tan pronto como reciben un valor  $v < \beta$ . La situación de poda se detecta en el momento en que un nodo satisface  $\alpha \geq \beta$  y ello implica obviar la exploración del sub-árbol del siguiente nodo hijo, y devolver  $\beta$  si se trata de un nodo MAX o bien  $\alpha$  si se trata de un nodo MIN. En caso de que todos los nodos hijos hayan sido explorados sin que se haya producido ninguna poda se devolverá  $\alpha$  si se trata de un nodo MAX o bien  $\beta$  si es un nodo MIN. Así, si se da el caso de que no se produce ninguna poda el algoritmo se comporta de forma idéntica a MINIMAX.

```

Algoritmo  $\alpha - \beta(n, \alpha, \beta)$  {
  Si (Es_HOJA( $n$ )) Devolver  $e(n)$ ;
  Sino {
    Si (Es_MAX( $n$ )) {
      Para cada  $s \in$  SUCESORES( $n$ ) Hacer{
         $\alpha \leftarrow \max \{\alpha, \alpha - \beta(s, \alpha, \beta)\};$ 
        Si ( $\alpha \geq \beta$ ) { Devolver  $\beta$  };
        Si (ES_ULTIMO( $s$ )) { Devolver  $\alpha$  };
      }
    }
    Sino {
      Para cada  $s \in$  SUCESORES( $n$ ) Hacer{
         $\beta \leftarrow \min \{\beta, \alpha - \beta(s, \alpha, \beta)\};$ 
        Si ( $\alpha \geq \beta$ ) { Devolver  $\alpha$  };
        Si (ES_ULTIMO( $s$ )) { Devolver  $\beta$  };
      }
    }
  }
}

```

**Figura 2.17.** Algoritmo de poda  $\alpha - \beta$ .



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

árbol que se muestra en la Figura 2.20a, los límites elegidos para la ventana inicial (10, 11) conducen a una *cota superior*, 9, del valor obtenido por MINIMAX, que sería 8. Esto se debe a que  $\alpha_0 = 10$  es *demasiado alto* y por tanto induce un corte que deja sin explorar la hoja etiquetada con el valor MINIMAX.

De manera similar, en la Figura 2.20b elegimos los límites (6, 7) que se propagan desde la raíz, pero las hojas del sub-árbol izquierdo no permiten actualizar el valor  $\beta$  de su nodo padre, que es de tipo MIN. Al devolver hacia arriba el valor 7 se actualiza el valor  $\alpha$  del nodo raíz y se induce un corte. El valor finalmente devuelto es 7, que es una *cota inferior* del valor MINIMAX, esto es, 8. Esto se debe a que  $\beta_0$  es *demasiado bajo*, con lo que resulta imposible que la hoja con el valor MINIMAX pueda actualizar el valor  $\beta$  de su nodo padre.

En estos dos casos, los valores iniciales cumplen  $\alpha_0 = \beta_0 - 1$ , lo cual se denomina búsqueda con *ventana nula*, en contraposición a la búsqueda clásica con *ventana infinita*.

**Algoritmo MTD-f** Una única llamada a un esquema de ventana nula como el que acabamos de describir solamente nos aporta información acerca de la cota (inferior o superior) del valor MINIMAX, pero no nos proporciona dicho valor (menos búsqueda implica menos información). Sin embargo, dichas cotas nos permiten modificar los límites iniciales  $\alpha_0$  y  $\beta_0$  para sucesivas llamadas. Con esta filosofía surge el algoritmo MTD-f (*Memory-enhanced Test Driver*) (véase la Figura 2.21). Partiendo de una estimación inicial  $f$  del valor MINIMAX se inicializan las cotas inferior  $f^-$  y superior  $f^+$  a  $-\infty$  y  $+\infty$  respectivamente. A continuación se entra en un ciclo de ajuste de dichas cotas que termina cuando la cota inferior iguala o supera la superior. Dicho ajuste se basa en una llamada al procedimiento  $\alpha - \beta$  con  $\alpha = \beta - 1$ , es decir, una búsqueda de ventana nula, que devuelve el valor  $g$ . Si  $g < \beta$  debemos ajustar la cota superior a dicho valor y en caso contrario debemos ajustar la cota inferior. El valor  $g$  sirve también para decidir el valor de  $\beta$  (y por tanto de  $\alpha = \beta - 1$ ) para la siguiente iteración. El número de iteraciones a realizar dependerá de lo lejos que esté la estimación inicial  $f$  del verdadero valor MINIMAX del nodo raíz.

Veamos qué ocurre en el ejemplo sencillo de la Figura 2.20. Ahora suponemos que el valor inicial es  $f = 10$ . Inicialmente los límites son  $f^- = -\infty, f^+ = \infty$ . Al inicializar  $g = f = 10$ , entramos en

**Algoritmo MTD-f( $n, f, l$ ) {**

```

 $g \leftarrow f;$ 
 $f^+ \leftarrow +\infty;$ 
 $f^- \leftarrow -\infty;$ 
Repetir {
    Si ( $g = f^-$ )  $\beta \leftarrow g + 1$ ; Sino  $\beta \leftarrow g$ ;
     $g \leftarrow \alpha - \beta M(n, \beta - 1, \beta, l)$ ;
    Si ( $g < \beta$ )  $f^+ \leftarrow g$ ; Sino  $f^- \leftarrow g$ ;
Hasta ( $f^- \geq f^+$ );
Devolver  $g$ ; }
```

Figura 2.21. Algoritmo MTD-f.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

---

```

Algoritmo BACKTRACKING( $G, a_i, S$ ) {
    Si ( $i = K$ ) Devolver  $S$ ;
     $j \leftarrow 1$ ;
    Repetir {
         $(S, f) \leftarrow \text{COMPONER}(G, S, v_{a_i} \leftarrow b_{ij})$ ;
        Si ( $f \neq \text{false}$ ) {
             $S \leftarrow \text{BACKTRACKING}(G, a_{i+1}, S)$ ;
            Si ( $S \neq \emptyset$ ) Devolver  $S$ ;
        }
         $j \leftarrow j + 1$ ;
    } Hasta ( $j = |D_{a_i}|$ );
    Devolver  $\emptyset$ ;
}

```

---

**Figura 2.26.** Algoritmo BACKTRACKING para CSPs.

argumento el grafo  $G$ , el índice de la primera variable  $a_1$  y una solución parcial vacía  $S \leftarrow \emptyset$ . En el  $i$ -ésimo nivel de recursión se intenta instanciar la variable  $v_{a_i} \in V$  con el primer valor  $b_{ij} \in D_{a_i}$ , en este caso el  $j$ -ésimo, con el que no se incumple ninguna de las restricciones del problema (aristas de  $E$ ). Esta comprobación se realiza en la función COMPONER. Si la instanciación es posible, entonces devuelve  $S \leftarrow S \cup (a_i \leftarrow b_{ij})$  y también devuelve el flag  $f \leftarrow \text{false}$ . En caso contrario, devuelve el  $S$  actual y  $f \leftarrow \text{true}$ . Si la composición ha sido posible, es hora de analizar la siguiente variable  $v_{a_{i+1}}$  para intentar seguir extendiendo  $S$ . Para ello se activa el siguiente nivel de recursión llamando de nuevo a BACKTRACKING. Hay que tener en cuenta que a la vuelta de esta llamada recursiva devuelve la primera solución factible encontrada o  $\emptyset$  si no encuentra ninguna. En caso de que  $S \neq \emptyset$ , devuelve  $S$  hacia la llamada anterior. En caso contrario, se constata que con el valor actual de la variable no se ha podido completar una solución factible y por lo tanto debemos probar con el siguiente valor. Si llegamos al final del dominio sin poder completar una solución factible en las llamadas subsiguientes entonces tenemos que devolver  $S \leftarrow \emptyset$ , para que la llamada anterior cambie el valor de la variable correspondiente.

**Árbol de búsqueda** El algoritmo que acabamos de ver implementa una búsqueda en profundidad en un *árbol de búsqueda* con tantos niveles como variables más uno (el correspondiente al nodo raíz) y tantos descendientes por nivel como valores tenga el dominio de cada variable. Precisamente, el orden de selección de variables permite ordenar los niveles (primero el más importante) y el orden de selección de valores permite ordenar los descendientes de cada nivel (primero el más a la izquierda). En la Figura 2.27 mostramos los cuatro primeros niveles (serían cinco si incluyésemos el nodo raíz) del árbol de búsqueda asociado a nuestro ejemplo de los crucigramas, de acuerdo con los ordenes de selección expuestos anteriormente. En el primer nivel examinamos la variable  $h_1$ , que aparece en 6 restricciones y le asignamos el valor RESTRICCIÓN, implicado en 7 restricciones. La siguiente variable es  $v_1$ , que aparece en 5 restricciones; se le asigna el valor PROGRAMA. Le sigue  $v_6$  a la que se le



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

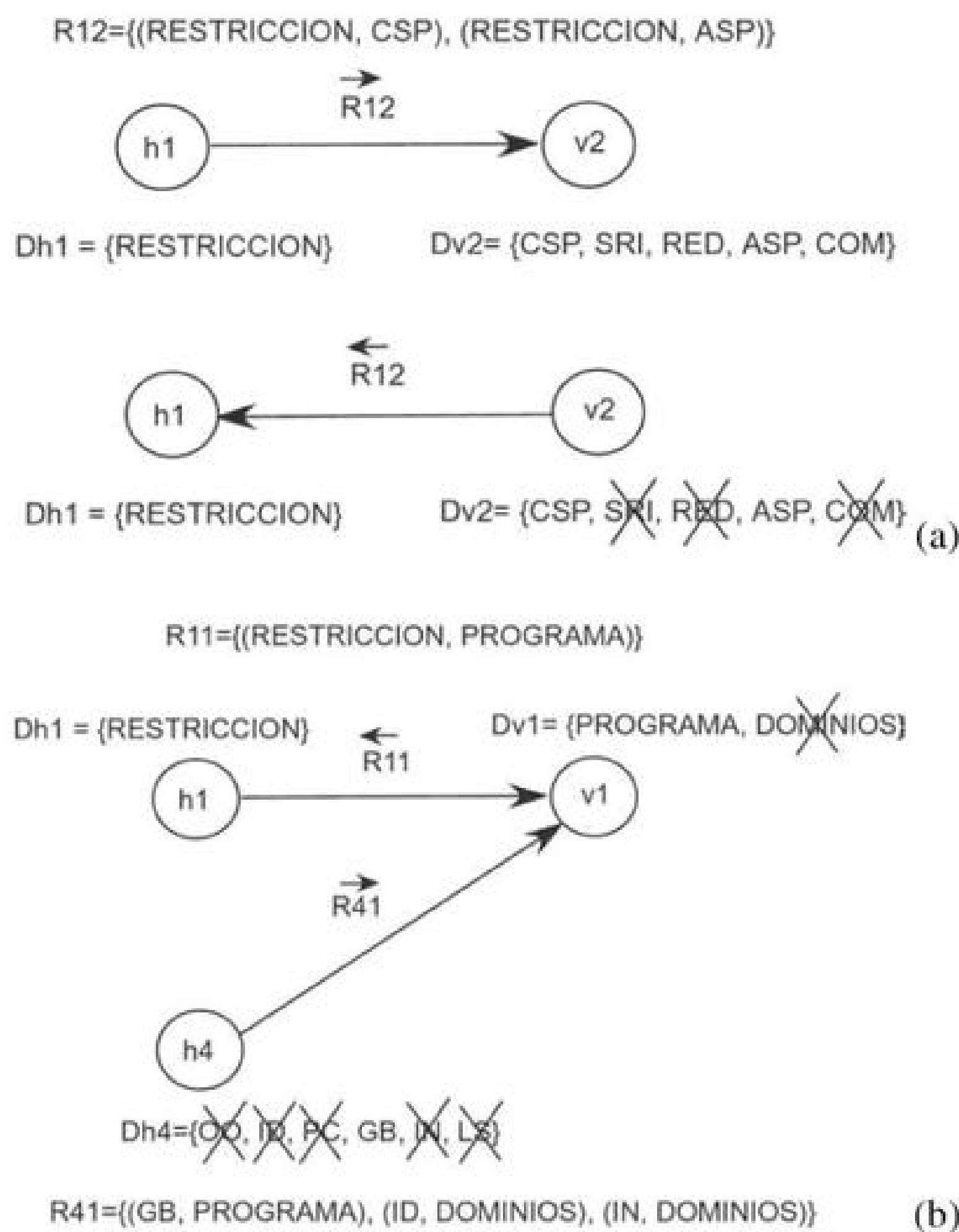


You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

cuando detectemos un valor que causa inconsistencia como  $\hat{a}$  es borrarlo de su dominio para que no dé lugar a *backtrackings* innecesarios.

Veamos un ejemplo concreto en nuestro problema del crucigrama (Figura 2.29a). Supongamos que estamos comprobando la consistencia en la restricción  $R_{12}$  (entre  $h_1$  y  $v_2$ ). Si contemplamos la propiedad en el sentido  $h_1 \rightarrow v_2$ , veremos que para el único valor de  $D_{h_1}$  que es RESTRICCIÓN, existe un par de valores de  $D_{v_2}$  (CSP y ASP) que hacen que la restricción se cumpla. Por esa razón no procede eliminar RESTRICCIÓN de  $D_{h_1}$ .

En cambio, cuando contemplamos la propiedad de consistencia en sentido  $v_2 \rightarrow h_1$ , encontramos que precisamente para todos los valores de  $D_{v_2}$  distintos de CSP y ASP no es posible verificar la misma restricción  $R_{12}$  y, por tanto, estos valores deben ser borrados de  $D_{v_2}$ . Así, vemos que la propiedad de consistencia de arista es una *propiedad direccional* y, por tanto, en un grafo de restricciones, cada arista bi-direccional debe desdoblarse en dos aristas uni-direccionales de cara a comprobar la consistencia.



**Figura 2.29.** Inconsistencia de arista. (a) La consistencia de arista es una propiedad uni-direccional. (b) Propagación de inconsistencias.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

no ocurre lo mismo con el algoritmo MTD-f, que se describe en el informe técnico de Plaat y otros colegas [Plaat and de Bruin, 1994] y es una variación de un test propuesto por Pearl en [Pearl, 1980] y descrito en su libro [Pearl, 1984].

Finalmente, el texto de Tsang [Tsang, 1993] realiza un recorrido exhaustivo por las técnicas y formalismos de satisfacción de restricciones, explorando las diversas variantes o alternativas de resolución. En él se encontrarán referencias detalladas a todos los algoritmos clásicos y modernos de satisfacción de restricciones. Aunque está agotado pueden obtenerse copias a través de la página web del autor. Algunos resultados recientes sobre resolución de crucigramas se pueden encontrar en [Littman *et al.*, 2002].



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

## Sistemas de programación lógica: el lenguaje PROLOG

**Programación lógica. El lenguaje PROLOG** La programación lógica es un paradigma de programación en el que un programa es un conjunto de fórmulas lógicas y su ejecución consiste en realizar una demostración. Así, un sistema de programación lógica es una realización práctica de un lenguaje de programación que permite formalizar conocimiento, demostrar hechos y responder a preguntas empleando los algoritmos que hemos estado viendo en el apartado anterior. De entre los sistemas de programación lógica existentes, PROLOG<sup>1</sup> (acrónimo de *PROgramming in LOGic*) es con mucho el más difundido en la actualidad. Dicho lenguaje se basa en un demostrador que funciona mediante resolución. En aras de la eficiencia, se introduce una serie de simplificaciones tanto en el lenguaje lógico empleado para definir la base de conocimientos como en la estrategia de control del algoritmo de resolución.

**Formalización del conocimiento en PROLOG** La limitación más destacada de PROLOG con respecto a la lógica de primer orden es que en este lenguaje sólo se pueden definir *cláusulas de Horn*. Una cláusula de este tipo es aquélla que como máximo tiene un átomo sin negar, por ejemplo, la cláusula 1 de la base de conocimientos de la página 63:

$$\neg\text{interesante}(a_1) \vee \neg\text{puedoAsistir}(a_1) \vee \text{aconsejable}(a_1)$$

Esta simplificación se introduce porque la lógica de las cláusulas de Horn es decidible, al contrario de lo que ocurre con la lógica de primer orden completa.

Además de esta limitación, hay algunas diferencias puramente sintácticas entre la notación de la lógica de primer orden y la de PROLOG. En éste, las cláusulas se expresan en forma de implicador, de manera que  $p_1 \vee \neg p_2 \vee \neg p_3 \vee \dots \vee \neg p_n$  se transforma en  $p_2 \wedge p_3 \wedge \dots \wedge p_n \rightarrow p_1$ . Nótese que la restricción de que haya como máximo un átomo positivo significa que en el consecuente del implicador sólo habrá un elemento. En notación PROLOG el implicador se pone “al revés”, es decir, primero el consecuente y luego el antecedente separados por los símbolos `: -` (de modo que  $q : -p \equiv p \rightarrow q$ ). En terminología PROLOG, la parte izquierda es la *cabeza* de la cláusula y la parte derecha el *cuerpo*. La conectiva  $\wedge$  se escribe en PROLOG como una coma, y las cláusulas terminan con un punto. Una última diferencia es que en PROLOG las variables se escriben en mayúsculas y las constantes en minúsculas, al contrario de la convención que se suele usar habitualmente en lógica. Así, la base de conocimientos de la pág. 63 se expresaría en PROLOG de la siguiente forma:

- ```

1  aconsejable(A) :- interesante(A), puedoAsistir(A) .
2  interesante(A) :- materia(A, M), gusta(M) .
3  puedoAsistir(A) :- horario(A, H), libre(H) .
4a horario(A, mañana) :- curso(A, tercero) .
4b horario(A, tarde) :- curso(A, tercero) .
5a materia(robotica, ia) .
5b materia(robotica, programacion) .

```

---

<sup>1</sup> Hay varias implementaciones de PROLOG disponibles. Por ejemplo, SWI-PROLOG es un intérprete PROLOG con licencia LGPL que puede conseguirse en <http://www.swi-prolog.org/>.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

---

**Algoritmo** ENCADENAMIENTO\_ADELANTE(*baseConoc*, *pregunta*) {

*deducido*  $\leftarrow \emptyset$

**Repetir** {

**Para cada** regla  $r = p_1 \wedge \dots \wedge p_n \rightarrow q \in \text{baseConoc}$

**Para cada**  $p'_1 \wedge \dots \wedge p'_n \in \text{baseConoc}$

        tal que  $\theta \leftarrow \text{UNIFICA}(p_1 \wedge \dots \wedge p_n, p'_1 \wedge \dots \wedge p'_n) \neq \text{fallo}$  {

$q' \leftarrow \text{SUST}(\theta, q)$

*baseConoc*  $\leftarrow \text{baseConoc} \cup q'$

$\rho \leftarrow \text{UNIFICA}(q', \text{pregunta})$

**si**  $\rho \neq \text{fallo}$  **devolver**  $\rho$  **fsi**

        }

    } **Hasta** *deducido* =  $\emptyset$

**Devolver** *falso*

}

---

**Figura 3.6.** Algoritmo de encadenamiento hacia delante.

---

**Algoritmo** ENCADENAMIENTO\_ATRAS(*baseConoc*, *pregunta*,  $\theta$ ) {

*respuesta*  $\leftarrow \emptyset$

**Si** *pregunta* =  $\emptyset$  **Devolver**  $\emptyset$

$q' \leftarrow \text{SUST}(\theta, \text{PRIMERO}(\text{pregunta}))$

**Para cada** regla  $r = p_1 \wedge \dots \wedge p_n \rightarrow q \in \text{baseConoc}$  {

$\theta' \leftarrow \text{UNIFICA}(q, q')$

**Si**  $\theta' \neq \text{fallo}$  {

*pregunta'*  $\leftarrow p_1 \wedge \dots \wedge p_n \wedge \text{RESTO}(\text{pregunta})$

*respuesta*  $\leftarrow \text{ENCADENAMIENTO\_ATRAS}(\text{baseConoc}, \text{pregunta}', \text{COMP}(\theta, \theta'))$

    }

}

**Devolver** *respuesta*

}

---

**Figura 3.7.** Algoritmo de encadenamiento hacia atrás.

## CLIPS: un entorno de desarrollo de sistemas basados en reglas

**Desarrollo de sistemas basados en reglas** Existe un elevado número de entornos (*shells*) y lenguajes de programación para el desarrollo de sistemas basados en reglas. El mismo PROLOG, por ejemplo, podría considerarse como un lenguaje de este estilo. De entre ellos, uno de los más conocidos es CLIPS.



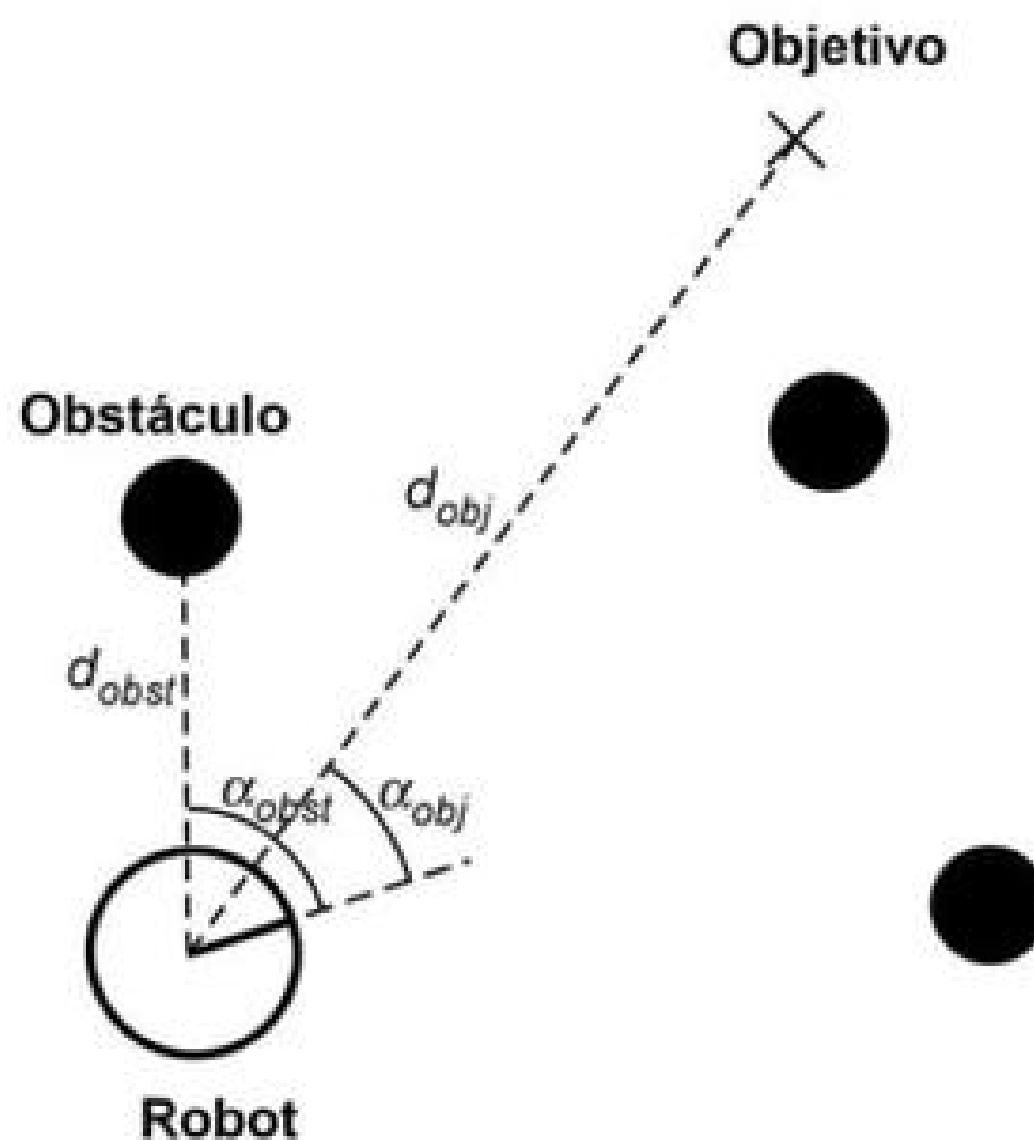
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



**Figura 3.8.** Problema de control de un robot móvil.

veremos, el control de la velocidad se realizará mediante un sistema basado en reglas, formuladas a partir de conceptos imprecisos como “ángulo-a-obstáculo pequeño” o “distancia-a-objetivo grande”.

## Conjuntos fuzzy

**Definición** El concepto de *conjunto fuzzy* es una generalización del concepto de *conjunto “clásico”*. En la teoría clásica un conjunto se define enumerando sus elementos o bien especificando la condición que deben cumplir, de manera que cada elemento del universo de discurso o bien pertenezcan o bien no pertenezcan al conjunto. Por el contrario, un conjunto *fuzzy* se define mediante una *función de pertenencia*  $\mu$ , que asocia a cada elemento del universo de discurso un *grado de pertenencia* en el intervalo  $[0, 1]$ . Esta forma de definir conjuntos es más apropiada que la clásica para modelar conceptos imprecisos como por ejemplo “joven”, “adulto” o “anciano”, ya que parece evidente que un individuo con 20 años es “joven” [es decir,  $\mu_{joven}(20) = 1$ ], pero el caso no es tan claro para alguien de 31 años. En lógica difusa podemos considerar  $\mu_{joven}(31) = 0,5$ , que no tiene equivalente en la teoría “clásica”. En el ámbito de la teoría *fuzzy* se suele utilizar el término *crisp* para denotar un conjunto “clásico”.

Volviendo al ejemplo del control del robot, en la Figura 3.9 puede observarse la diferencia entre modelar los conceptos de distancia “próxima”, “cercana”, “media” y “grande” empleando conjuntos *crisp* y conjuntos difusos. Como puede verse, el primer enfoque no es apropiado en este contexto, ya que no parece demasiado lógico que una distancia de 90 cm se defina arbitrariamente como cercana pero no media, y con solo un cm más pase a ser media y deje absolutamente de ser cercana. Aun siendo también arbitrarias, las funciones de pertenencia  $\mu$  permiten expresar una *gradación* que modela mejor los conceptos de este tipo. Además, un elemento del dominio puede pertenecer a distintos conjuntos difusos en distintos grados. Así, por ejemplo, según la Figura 3.9, una distancia de 140



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

**Diseño de variables lingüísticas en sistemas fuzzy** Una *variable lingüística* es aquélla cuyos valores son conjuntos difusos. En el ejemplo anterior, la variable sería *distancia al objetivo* y podría tomar cualquiera de los valores *pequeña, media, grande, muy grande, ...*. Cada uno de estos valores es lo que se denomina un *término lingüístico*. Los sistemas difusos utilizan variables lingüísticas como variables de entrada y salida. Por ello, el primer paso en el diseño de un sistema difuso consiste en identificar dichas variables, su dominio, los conjuntos difusos que las forman y los modificadores permitidos. Volviendo al ejemplo inicial del controlador difuso para un robot móvil, a partir de las especificaciones del problema podemos ver que hay cuatro variables de entrada y dos de salida. La Figura 3.12 muestra los datos de cada una de las variables y los conjuntos difusos definidos a partir de ellos. La elección del número de conjuntos difusos y la forma de la función de pertenencia para cada uno es decisión del diseñador del sistema, que empleará para ello su conocimiento del dominio y su experiencia en diseños previos. Como heurística general, no obstante, se puede indicar que los conjuntos difusos deben solaparse entre sí de modo que no haya puntos del dominio que no pertenezcan a ningún conjunto.

**Operaciones con conjuntos fuzzy** Básicamente, las operaciones que se pueden realizar con conjuntos fuzzy son extensiones de las de la teoría clásica de conjuntos. Las fundamentales son, pues, la *unión*, la *intersección* y el *complemento*. En este texto utilizaremos como implementación de dichas operaciones la siguiente:

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <b>Complemento</b>  | $\mu_{A'}(x) = 1 - \mu_A(x)$                     |
| <b>Unión</b>        | $\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$ |
| <b>Intersección</b> | $\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$ |

La Figura 3.13 muestra el resultado de realizar estas operaciones. No obstante, máx y mín no son las únicas formas de implementar  $\mu_{A \cap B}$  y  $\mu_{A \cup B}$ . En general, cualquier operador que cumpla las condiciones necesarias para ser lo que se denomina una *t-conorma* o una *t-norma* puede servir para implementar la unión e intersección de conjuntos fuzzy, respectivamente, aunque aquí no entraremos en la definición formal de dichas condiciones.

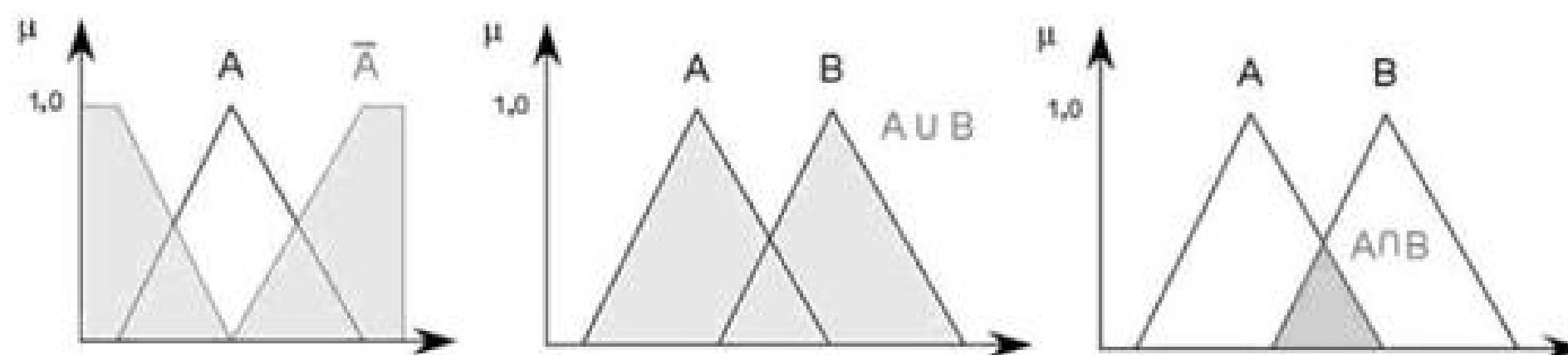


Figura 3.13. Operaciones con conjuntos difusos.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

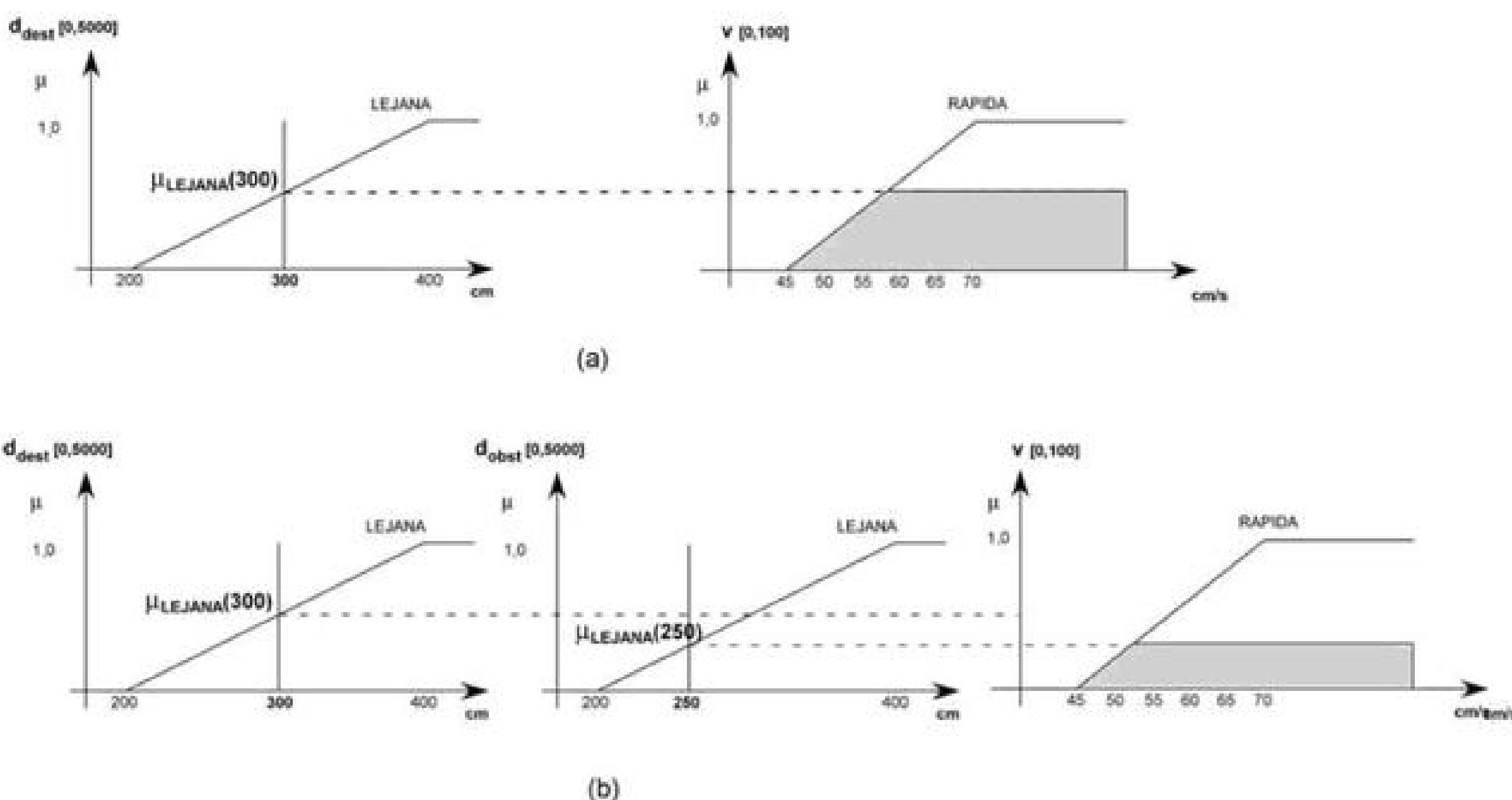


You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

$$\begin{bmatrix} 250 & 300 & 350 & 400 & 450 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \circ \begin{bmatrix} 40 & 50 & 60 & 70 \\ 0 & 0,2 & 0,25 & 0,25 \\ 0 & 0,2 & 0,5 & 0,5 \\ 0 & 0,2 & 0,6 & 0,75 \\ 0 & 0,2 & 0,6 & 1 \\ 0 & 0,2 & 0,6 & 1 \end{bmatrix} = \\
 = \begin{bmatrix} 40 & 50 & 60 & 70 \\ 0 & 0,2 & 0,5 & 0,5 \end{bmatrix}$$

Puede observarse que el conjunto resultante no es más que la fila de la matriz  $A \rightarrow B$ , que se compone con el único punto del dominio  $x$  para el que  $\mu_{A'}(x) > 0$ . Como en este caso,  $A \rightarrow B$  se ha implementado con el implicador de Mamdani, esta fila no es más que  $\min_{y \in Y} \{\mu_A(x), \mu_B(y)\}$ . Este mínimo para todos los puntos del dominio puede verse gráficamente en la Figura 3.15a. En general, si  $*$  representa el operador con el que se ha implementado la implicación difusa, y  $x$  es el valor del *singleton*, la función de pertenencia del conjunto resultante para un punto del dominio  $y$  será  $\mu_A(x) * \mu_B(y)$ .

En una regla con múltiples antecedentes  $A_1, A_2, \dots, A_n$  y múltiples entradas de tipo *singleton*,  $x_1, x_2, \dots, x_n$ , la expresión anterior se convierte en  $\min\{\mu_{A_1}(x_1), \dots, \mu_{A_n}(x_n)\} * \mu_B(y)$  cuando los antecedentes están unidos con  $\wedge$  (Figura 3.15b) o en  $\max\{\mu_{A_1}(x_1), \dots, \mu_{A_n}(x_n)\} * \mu_B(y)$  cuando están unidos con  $\vee$  (suponiendo que se implementa  $\wedge$  con  $\min$  y  $\vee$  con  $\max$ ).



**Figura 3.15.** Inferencia difusa con entradas *singleton*. (a) Regla con un solo antecedente (b) Regla con múltiples antecedentes.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

**Entrenamiento como ajuste supervisado de hiperplanos** Supongamos ahora que los pesos de una recta  $w_1$  y  $w_2$ , y su umbral  $\theta$  son desconocidos, pero que sabemos qué etiquetas van a tener los puntos. En el ejemplo que se muestra en la Figura 4.3, tenemos dos grupos de 5 puntos cada uno, de tal forma que los puntos etiquetados como 1 aparecen en gris claro mientras que los etiquetados como 0 aparecen en gris oscuro. Así pues, supongamos una situación inicial en donde  $w_1 = 1$ ,  $w_2 = 0$  y  $\theta = 7$ , esto es, la recta  $x_1 - 7 = 0$ . Evidentemente, dicha recta no separa correctamente ambos conjuntos de patrones. Así que la cuestión esencial en este punto estriba en proponer una *regla de actualización* de los pesos y del umbral (de los pesos en sentido amplio, ya que el umbral es un peso más) de forma que la recta se aproxime un poco más hacia una recta que permita separar los conjuntos.

Supongamos que  $d$  es la salida que *debería* proporcionar la neurona para un ejemplo dado, mientras que  $y$  es la salida que *realmente* obtiene la neurona con la configuración de pesos actuales. El error asociado a la neurona con dicha configuración es  $d - y$ . Así, la denominada “regla delta” estipula que el incremento/decremento del peso  $w_i$  depende de dicho error de la siguiente manera:

$$\Delta w_i = \eta(d - y)x_i \text{ o sea } w_i = w_i + \eta(d - y)x_i \quad (4.1)$$

donde  $d - y$  suele denominarse como  $\delta$ . Así, si la neurona acierta la clase, no hay que modificar nada (tenemos que  $\delta = 0$ ). Si por el contrario falla, a cada peso le sumamos o restamos (dependiendo del signo de  $\delta$ ) el valor de la entrada correspondiente, multiplicada por un factor de proporcionalidad,  $\eta$ . Como puede verse en la Figura 4.3, tras presentar cada uno de los ejemplos  $e_1, e_2, e_3$  y  $e_4$ , la recta

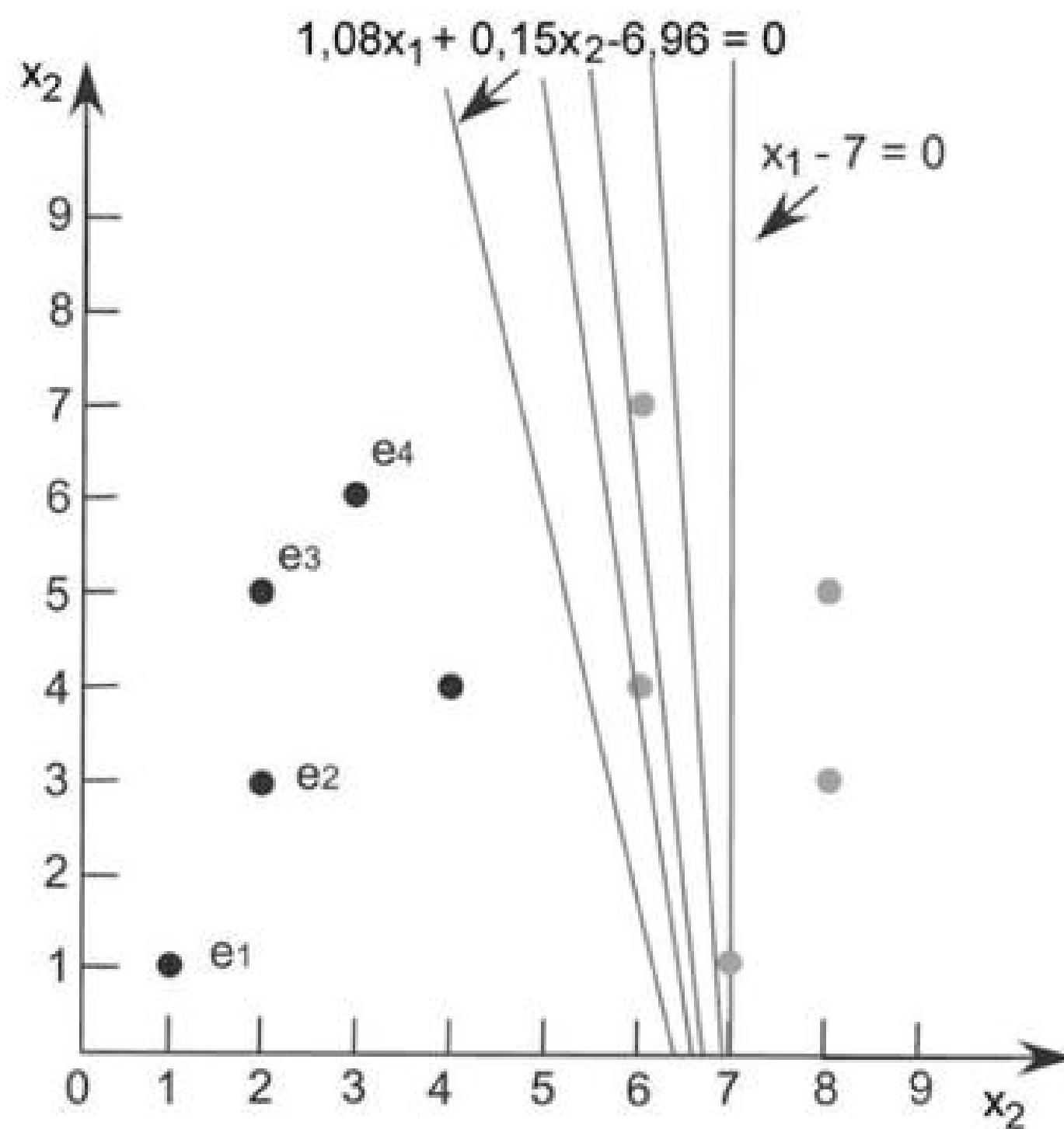


Figura 4.3. Convergencia de la regla delta a medida que se incluyen ejemplos de aprendizaje.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

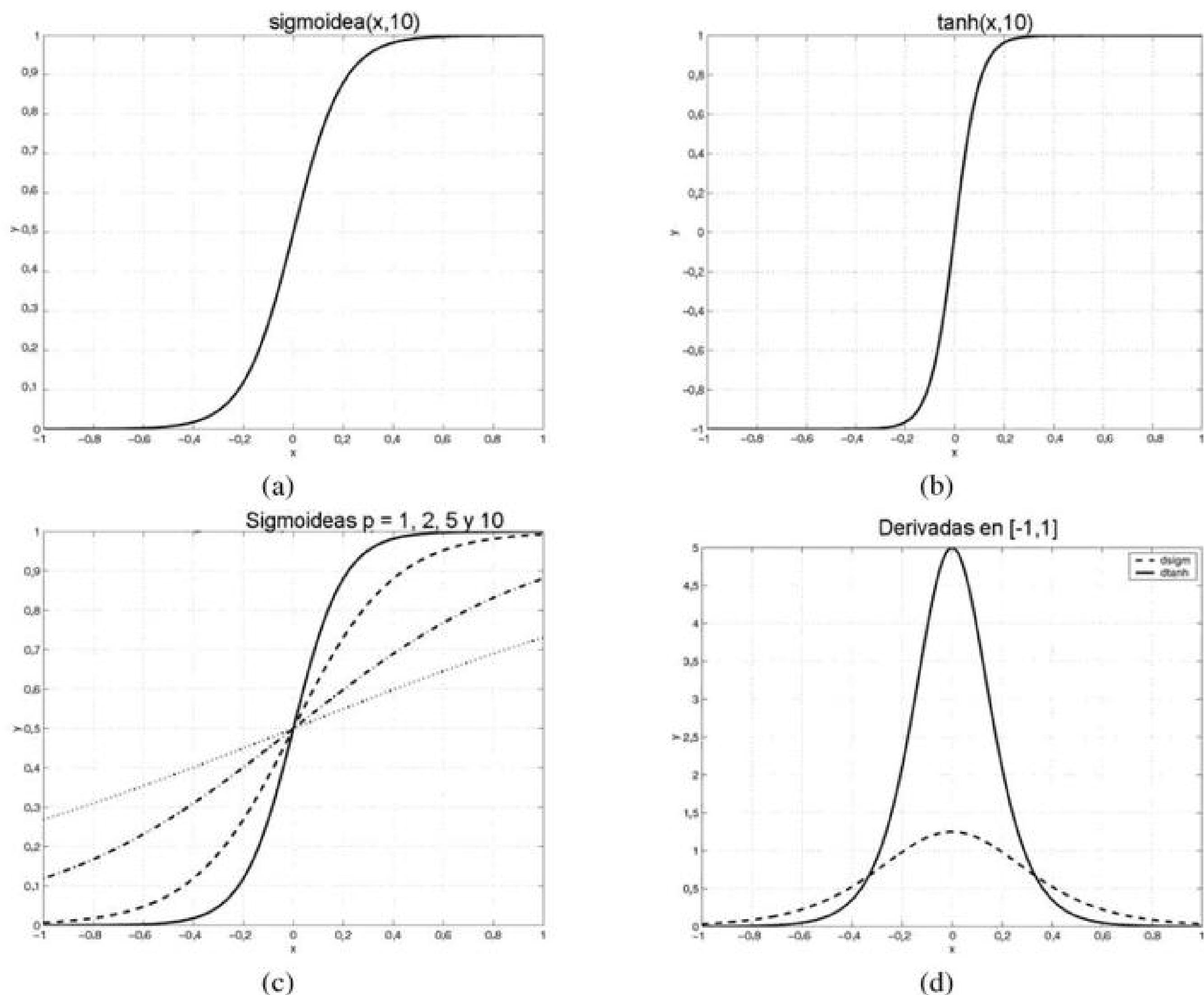


You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

es decir, la diferencia entre la salida deseada para esa neurona,  $d_k$ , y la obtenida,  $y_k$ , multiplicada por la derivada de la función de activación,  $f'(net_k)$ . Así, la regla de actualización de los pesos  $w_{jk}$  que ligan la capa oculta  $j$  con la capa de salida  $k$  sería la siguiente:

$$w_{jk} = w_{jk} + \eta \delta_k y_j \quad (4.7)$$

Si queremos aplicar la regla para actualizar los pesos  $w_{ij}$ , necesitamos saber cuáles serán los valores de los  $\delta_j$  correspondientes. La idea es que una neurona oculta de la capa  $j$  es responsable de una parte del error  $\delta_k$  de cada una de las neuronas a las que se conecta su salida. Por ello, el error cometido en la neurona de la capa  $j$  será la suma de los errores de las neuronas de la capa siguiente ponderados por los pesos que conectan con ellas [y de nuevo aparece el factor  $f'(net_j)$ ]:



**Figura 4.6.** Funciones de activación. Sigmoidea (a) y tangente hiperbólica (b). Ambas con  $p = 10$ . A medida que aumenta  $p$ , las funciones de activación (en este caso la sigmoidea) se aproximan a una funciones escalón (c). Derivadas de las funciones de activación (d).



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



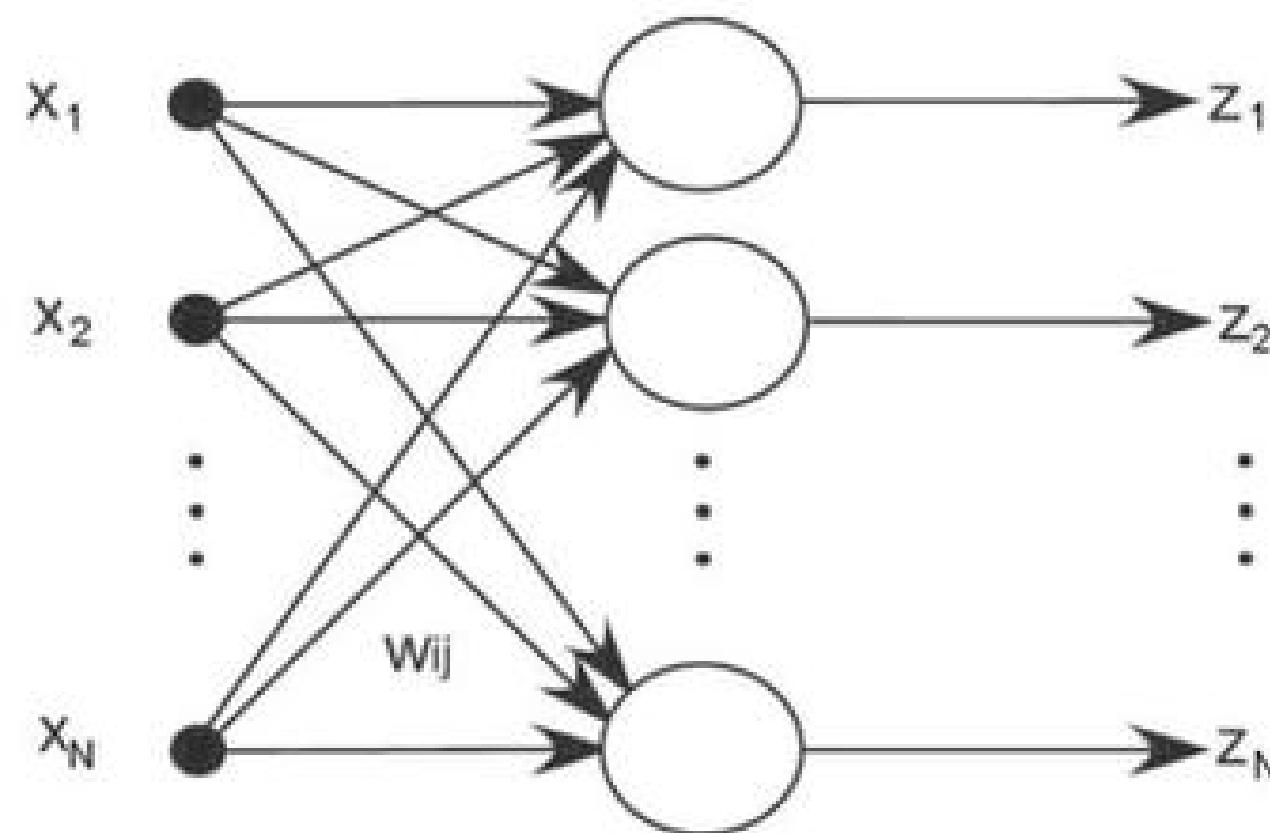
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



**Figura 4.13.** Memoria asociativa con umbralización.

Parece por tanto adecuado dar una salida u otra en función (no-lineal) de la acumulación de correlaciones (positiva o negativa) para la componente en cuestión. Veamos el funcionamiento de este mecanismo con un ejemplo de asociación de imágenes, dados los cuatro patrones de dimensión  $N = 4 \times 2 = 8$ , representados en la Figura 4.14, en donde el valor de intensidad 0 se codifica como  $-1$  y el valor 1 se codifica como  $1$ . Utilizando la matriz de pesos resultante, simétrica y de dimensión  $8 \times 8$ , en combinación con el umbralizado, la red resultante es capaz de recuperar cada uno de los patrones de entrada. Concretamente, para el patrón  $\mathbf{x}_1$ , el vector **net** resultante es:

$$\text{net} = \mathbf{W}\mathbf{x}_1 = \begin{bmatrix} 4 & -2 & 2 & 0 & 0 & -2 & -2 & 0 \\ -2 & 4 & 0 & 2 & -2 & 0 & 0 & -2 \\ 2 & 0 & 4 & -2 & -2 & 0 & 0 & -2 \\ 0 & 2 & -2 & 4 & 0 & -2 & -2 & 0 \\ 0 & -2 & -2 & 0 & 4 & -2 & 2 & 0 \\ -2 & 0 & 0 & -2 & -2 & 4 & 0 & 2 \\ -2 & 0 & 0 & -2 & 2 & 0 & 4 & -2 \\ 0 & -2 & -2 & 0 & 0 & 2 & -2 & 4 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -8 \\ 8 \\ -8 \\ 8 \\ -8 \\ 8 \\ -8 \\ 8 \end{bmatrix}$$

Lo cual implica que el umbralizado devolverá el patrón de entrada  $\mathbf{x}_1$ . Sin embargo, si mostramos  $\mathbf{x}'_1$ , una versión levemente distorsionada del patrón  $\mathbf{x}_1$  (solamente cambia la segunda componente), el **net** resultante es:

$$\text{net} = \mathbf{W}\mathbf{x}'_1 = \begin{bmatrix} 4 & -2 & 2 & 0 & 0 & -2 & -2 & 0 \\ -2 & 4 & 0 & 2 & -2 & 0 & 0 & -2 \\ 2 & 0 & 4 & -2 & -2 & 0 & 0 & -2 \\ 0 & 2 & -2 & 4 & 0 & -2 & -2 & 0 \\ 0 & -2 & -2 & 0 & 4 & -2 & 2 & 0 \\ -2 & 0 & 0 & -2 & -2 & 4 & 0 & 2 \\ -2 & 0 & 0 & -2 & 2 & 0 & 4 & -2 \\ 0 & -2 & -2 & 0 & 0 & 2 & -2 & 4 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -4 \\ 0 \\ -8 \\ 4 \\ -4 \\ 8 \\ -8 \\ 12 \end{bmatrix}$$



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

lo tanto, la existencia de una función de Lyapunov para ese sistema nos garantizaría la existencia de puntos de estabilidad del mismo, es decir, que en un determinado momento la secuencia de transiciones alcanza un punto fijo (el vértice que devuelve como respuesta). Sin embargo, dicha función debe ser monótona decreciente a medida que se produce una transición, y debe estar acotada. Hopfield definió la siguiente función de Lyapunov:

$$E(\mathbf{z}) = -\frac{1}{2} \mathbf{z}' \mathbf{W} \mathbf{z} = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N z_i z_j w_{ij} \quad (4.23)$$

Dado que la diagonal de  $\mathbf{W}$  es nula, esto es,  $w_{ii} = 0$ , podemos expresar dicha función en los siguientes términos:

$$E(\mathbf{z}) = -\frac{1}{2} \sum_{i=1}^N z_i \left( \sum_{j \neq i} z_j w_{ij} \right) \quad (4.24)$$

ya que,

$$E(\mathbf{z}) = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N z_i z_j w_{ij} = \quad (4.25)$$

$$= -\frac{1}{2} \sum_{i=1}^N z_i \left( \sum_{j=1}^N z_j w_{ij} \right) = \quad (4.26)$$

$$= -\frac{1}{2} \sum_{i=1}^N z_i \left( z_i w_{ii} + \sum_{j \neq i} z_j w_{ij} \right) = \quad (4.27)$$

$$= -\frac{1}{2} \sum_{i=1}^N z_i \left( \sum_{j \neq i} z_j w_{ij} \right) \quad (4.28)$$

Así, podemos especificar ( $\mathbf{z}$ ) en términos de la contribución de una componente  $z_q$ :

$$E(\mathbf{z}) = -z_q \frac{1}{2} \sum_{j \neq q} z_j w_{qj} - \frac{1}{2} \sum_{i \neq q} z_i \left( \sum_{j \neq i} z_j w_{ij} \right) \quad (4.29)$$

Supongamos que tenemos la transición  $\mathbf{z} \rightarrow \mathbf{z}'$ , en la que puede cambiar solamente la componente  $z_r$ . Entonces el incremento de energía  $\Delta E$  viene dado por:

$$\Delta E = E(\mathbf{z}') - E(\mathbf{z}) = -z_r' \frac{1}{2} \sum_{j \neq q} z_j w_{rj} + z_r \frac{1}{2} \sum_{j \neq q} z_j w_{rj} = (-z_r' + z_r) \frac{1}{2} \sum_{j \neq q} z_j w_{rj} \quad (4.30)$$

De lo que se deduce que si la neurona no se dispara o bien se dispara pero no cambia de valor, entonces la energía no varía. En definitiva, no se produce una transición a un nuevo vértice del hipercubo, ya que  $\mathbf{z}' = \mathbf{z}$ . Sin embargo, cuando  $z_r' = -z_r$  entonces *si* se produce un cambio de estado cuyo incremento asociado es:

$$\Delta E = z_r \sum_{j \neq q} z_j w_{rj} \quad (4.31)$$



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

esto es, la probabilidad de haber realizado la observación parcial de  $O_1 O_2 \dots O_t$  y de estar en el estado  $q_i$  en el instante  $t$ , dado el modelo  $\lambda$ . Así, cuando  $t = T$ ,  $\alpha_T(i)$  cuantifica la probabilidad de realizar la observación completa  $O_1 O_2 \dots O_T$  y acabar en el estado  $q_i$ . Si sumamos los  $\alpha_T(i)$  de todos los estados obtenemos la probabilidad de realizar la observación completa terminando en *alguno de los estados* de  $Q$  y eso es, en términos generales, una medida de  $P(O|\lambda)$ . Así pues:

$$P(O|\hat{\lambda}) = \sum_{i=1}^N \alpha_T(i) \quad (4.48)$$

Partiendo de la ecuación anterior hay que tener en cuenta que la evaluación de las  $\alpha_T(i)$  puede realizarse de forma incremental con  $t$ . Así, cuando  $t = 1$ , esto es, en el instante inicial, se cumple que:

$$\alpha_1(i) = \pi_i b_i(O_1) \text{ para } 1 \leq i \leq N \quad (4.49)$$

ya que la probabilidad de observar  $O_1$  habiendo llegado a  $q_i$  en  $t = 1$  es la probabilidad de que  $q_i$  sea el estado inicial multiplicada por la probabilidad de observar  $O_1$  desde  $q_i$ .

Por otro lado, para  $t = 1, 2, \dots, T - 1$  y  $1 \leq j \leq N$ , se cumple que:

$$\alpha_{t+1}(j) = \left[ \sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \quad (4.50)$$

ya que la probabilidad de alcanzar  $q_j$  en  $t + 1$  y observar  $O_{t+1}$  se puede factorizar en dos términos: (1) *alcanzar*  $q_j$  y (2) *observar*  $O_{t+1}$  desde  $q_j$ . El primer término se obtiene de considerar que para alcanzar  $q_j$  debemos haber alcanzado antes cualquiera de los estados  $q_i$  (incluido el propio  $q_j$ ) y de ellos realizar la transición de probabilidad  $a_{ij}$ . El segundo término se obtiene simplemente consultando  $b_j(O_{t+1})$ .

Como puede verse en la parte izquierda de la Figura 4.26, dada la secuencia  $ACA\dots$ ,  $\alpha_2(2)$  en nuestro HMM de ejemplo se calcula a partir de los  $\alpha_1(i)$ , siendo  $q_i$  un estado desde el que se puede alcanzar  $q_2$ , y de las probabilidades de transición  $a_{i2}$  (primer término) y de consultar  $b_2(C)$ , ya que  $O_2 = C$  (segundo término). No obstante, dado que el estado inicial del HMM es  $q_1$  con total probabilidad, y que la transición  $a_{12}$  tiene probabilidad nula, los  $\alpha_1(i)$  son también nulos y consecuentemente  $\alpha_2(2)$  también lo es.

De manera similar a las variables anteriores, denominadas *forward* por la forma en la que se calculan, se definen las variables *backward*  $\beta_t(i)$ , que cuantifican la probabilidad de la observación parcial  $O_t O_{t+1} \dots O_T$ , dado que estamos en  $q_t$  en el instante  $t$  y disponemos del modelo  $\lambda$ :

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \dots, O_T | q_t = i, \lambda) \quad (4.51)$$

esto es, la probabilidad de observar lo que queda de secuencia a partir de un estado determinado  $q_i$ . Así,  $\beta_1(i)$  es la probabilidad de observar la secuencia completa a partir del estado  $q_i$ . Si consideramos además  $\pi_i$ , esto es, la probabilidad de que  $q_i$  sea el estado inicial, y  $b_i(O_1)$ , la probabilidad de que en ese estado observemos el primer símbolo de la cadena, entonces resulta que  $P(O|\lambda)$  puede definirse también en términos de:

$$P(O|\lambda) = \sum_{i=1}^N \hat{\pi}_i b_i(O_1) \beta_1(i) \quad (4.52)$$



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

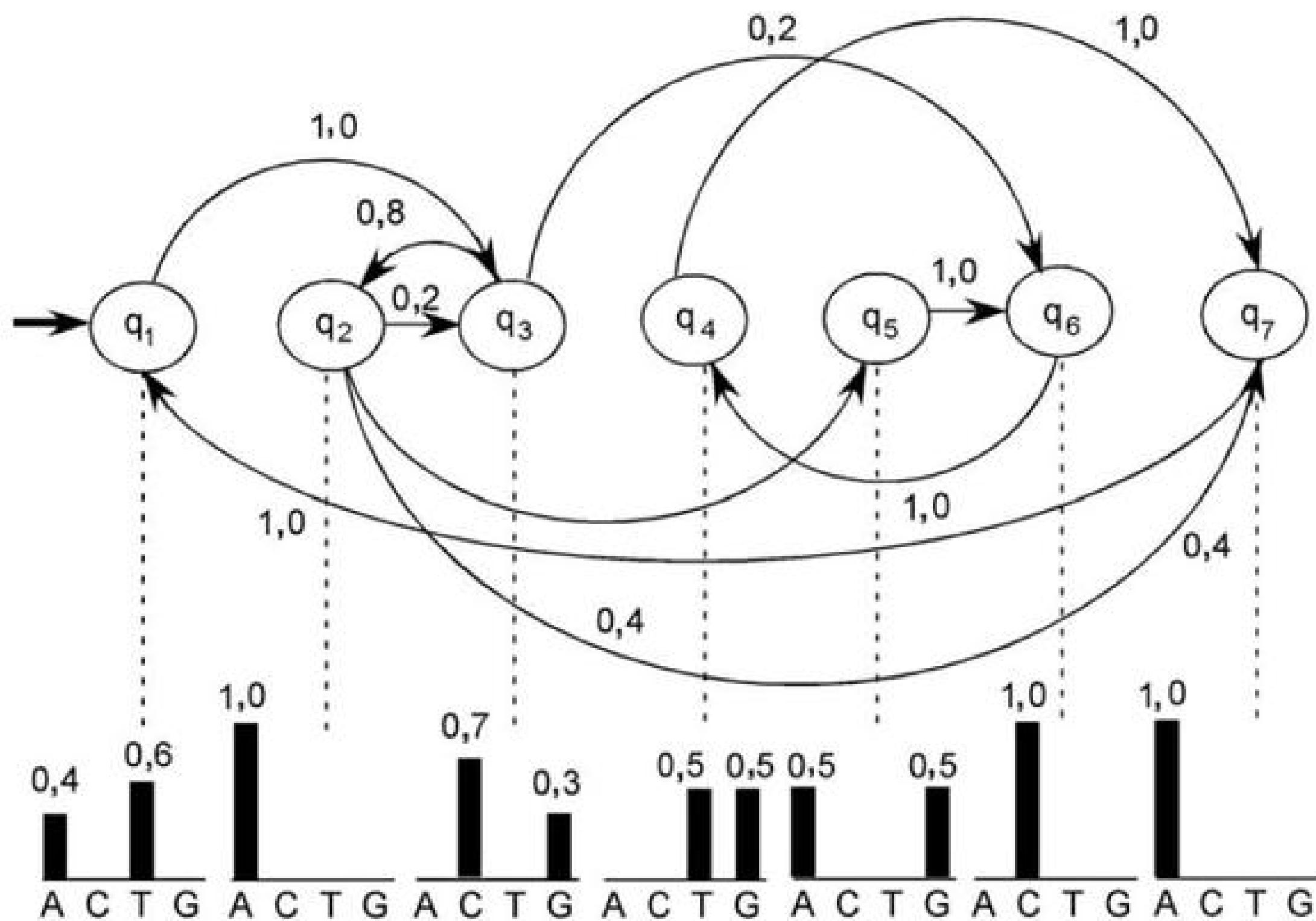


Figura 4.29. HMM alternativo partiendo de similares condiciones de inicialización.

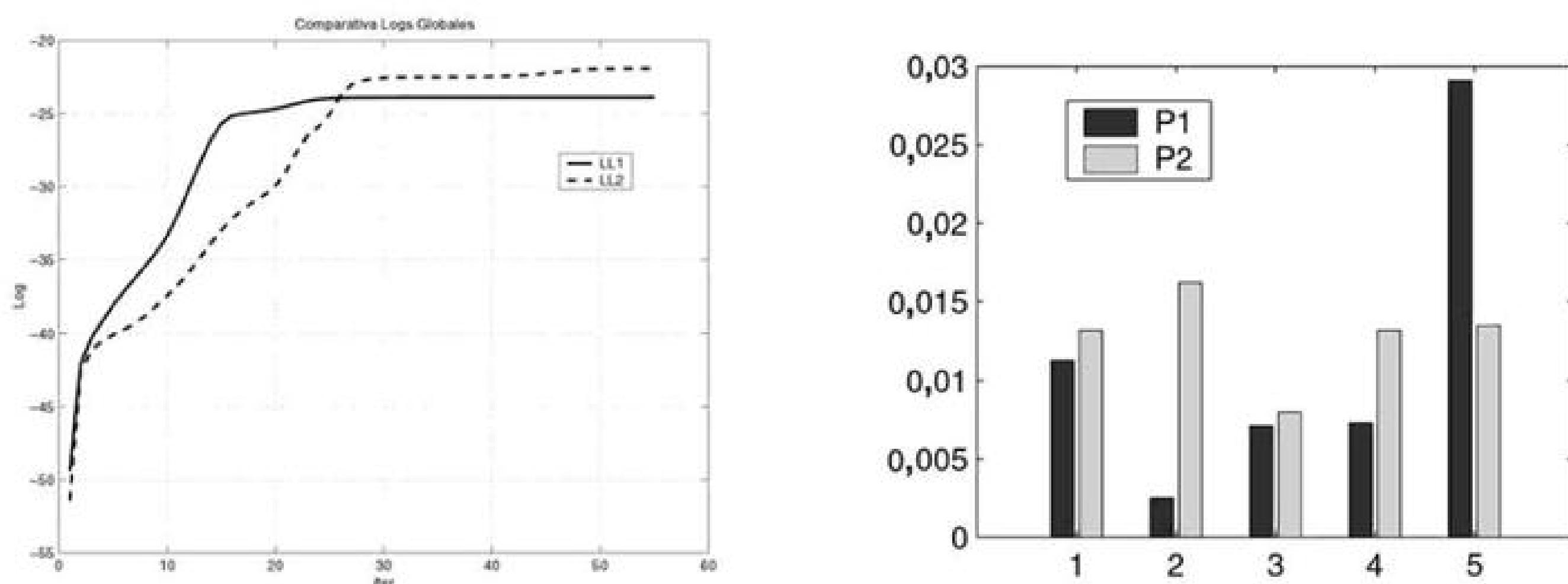


Figura 4.30. Comparativa de  $\log$  y  $P()$  para HMM1 y HMM2.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

| Cliente | Moroso | Antigüedad (años) | Ingresos (Eur./mes) | Trab.fijo | Conceder |
|---------|--------|-------------------|---------------------|-----------|----------|
| 1       | sí     | >5                | 600-1200            | sí        | sí       |
| 2       | no     | <1                | 600-1200            | sí        | no       |
| 3       | sí     | 1-5               | >1200               | sí        | no       |
| 4       | no     | >5                | >1200               | no        | sí       |
| 5       | no     | <1                | >1200               | sí        | sí       |
| 6       | sí     | 1-5               | 600-1200            | sí        | no       |
| 7       | no     | 1-5               | >1200               | sí        | sí       |
| 8       | no     | <1                | <600                | sí        | no       |
| 9       | no     | >5                | 600-1200            | no        | no       |
| 10      | no     | 1-5               | <600                | no        | no       |

**Figura 4.34.** Datos de créditos solicitados con anterioridad.

el valor del atributo correspondiente, hasta llegar a conjuntos en los que todos los ejemplos son de la misma clase, que serán los nodos hoja del árbol. Si se han utilizado todos los atributos y aun así no todos los ejemplos pertenecen a la misma clase, se etiqueta el nodo hoja con la clase mayoritaria en el conjunto.

El algoritmo más representativo de este tipo es el conocido como ID3. Este algoritmo emplea como heurística para evaluar el mejor atributo una medida estadística conocida como *ganancia de información*, que refleja en qué medida un atributo sirve para discriminar entre las distintas clases. Si un atributo es muy discriminante, generará grupos de ejemplos bastante homogéneos (de la misma clase) para cada uno de sus valores. Este comportamiento tenderá a generar árboles pequeños (con pocos niveles), ya que se seleccionan aquellos atributos que tienen mayor probabilidad de generar hojas (conjuntos totalmente homogéneos) en pocos pasos. El algoritmo se detalla en la Figura 4.35 (el cálculo de la ganancia de información  $G$  se explica a continuación).

**Entropía y ganancia de información** La teoría de la información nos proporciona una manera de formalizar el concepto de “grupos de ejemplos homogéneos”, a través del concepto de *entropía*. En este contexto, definiremos la entropía de un conjunto de ejemplos  $S$  como:

$$E(S) = \sum_{i \in C} -p_i \log_2 p_i$$

donde  $C$  es el conjunto de clases a las que pueden pertenecer dichos ejemplos y  $p_i$  es la probabilidad de que un ejemplo dado pertenezca a la clase  $i$ -ésima (normalmente ésta se aproxima mediante el número de ejemplos de dicha clase partido por el total de ejemplos). La entropía se expresa en bits, de ahí la base 2 del logaritmo. Esta medida se puede interpretar como la *incertidumbre* que tenemos sobre la clase a la que pertenecerá un ejemplo escogido al azar del conjunto  $C$ . Es sencillo comprobar que para conjuntos totalmente homogéneos (todos los ejemplos de la misma clase), la entropía será siempre 0 (consideraremos que  $0 \log 0$  es igual a 0).



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

Según esta perspectiva, y para los propósitos de esta discusión, la finalidad de un algoritmo evolutivo será la de maximizar (o minimizar) una función  $f : S \rightarrow \mathbb{R}$ , donde  $S$  es el espacio de búsqueda.

**Características de los algoritmos de computación evolutiva** Los algoritmos evolutivos se pueden utilizar como métodos de optimización de “caja negra”, ya que no necesitan más conocimiento sobre la función a optimizar que poder obtener muestras de la misma. De este modo, no es necesario que las funciones a tratar cumplan restricciones matemáticas específicas, como ocurre con otros métodos (por ejemplo, el requisito de derivabilidad en el descenso de gradiente), y ni siquiera es necesario tener las funciones en forma cerrada, sino que los valores pueden ser obtenidos, por ejemplo, de una simulación. Éste es, a la vez, el punto fuerte y el problema principal de este tipo de métodos: al no poseer conocimiento *a priori* sobre la función, son mucho menos eficientes que otros más específicos. Por ello, son apropiados cuando no se puede aplicar otros métodos, o bien cuando se combina con éstos otros algoritmos, formando métodos híbridos. Por otro lado, los algoritmos evolutivos son métodos de búsqueda global, es decir, se pueden emplear para optimizar funciones con múltiples máximos o mínimos locales (funciones multimodales).

## De la inspiración biológica al modelo computacional

**La inspiración biológica** En la naturaleza, las características físicas externas de cada individuo (lo que los biólogos denominan fenotipo) están codificadas de algún modo en el conjunto de sus genes (o genotipo). Según las teorías biológicas más aceptadas en la actualidad, expuestas aquí de manera muy simplificada, la evolución es el resultado de la interacción entre dos factores: la creación al azar de nueva información genética, y la evaluación y selección no azarosa de los individuos. Cuanto más adaptado está un individuo al entorno en el que vive, mayor es la probabilidad de que sobreviva

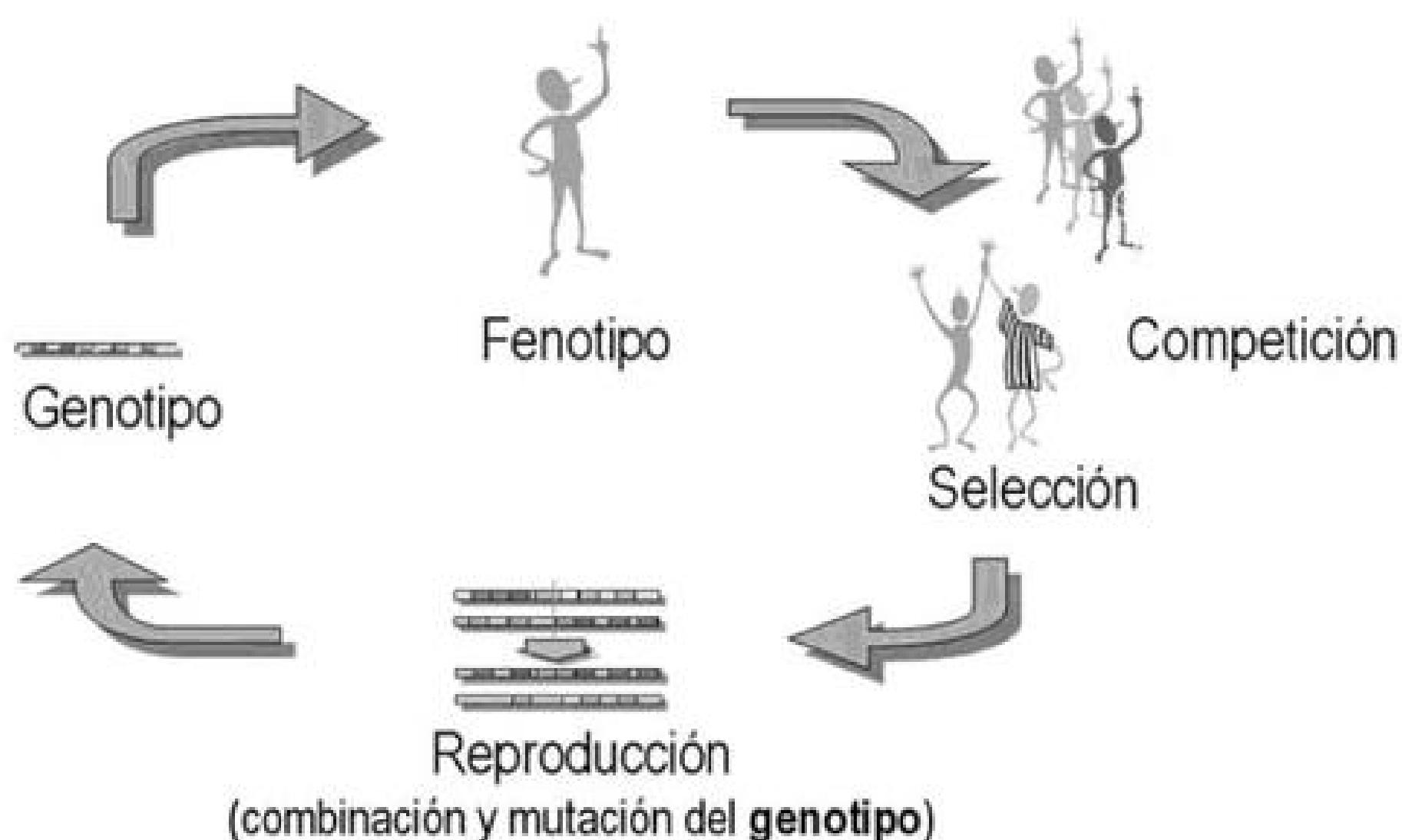


Figura 4.37. Modelo simplificado del ciclo evolutivo en biología.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

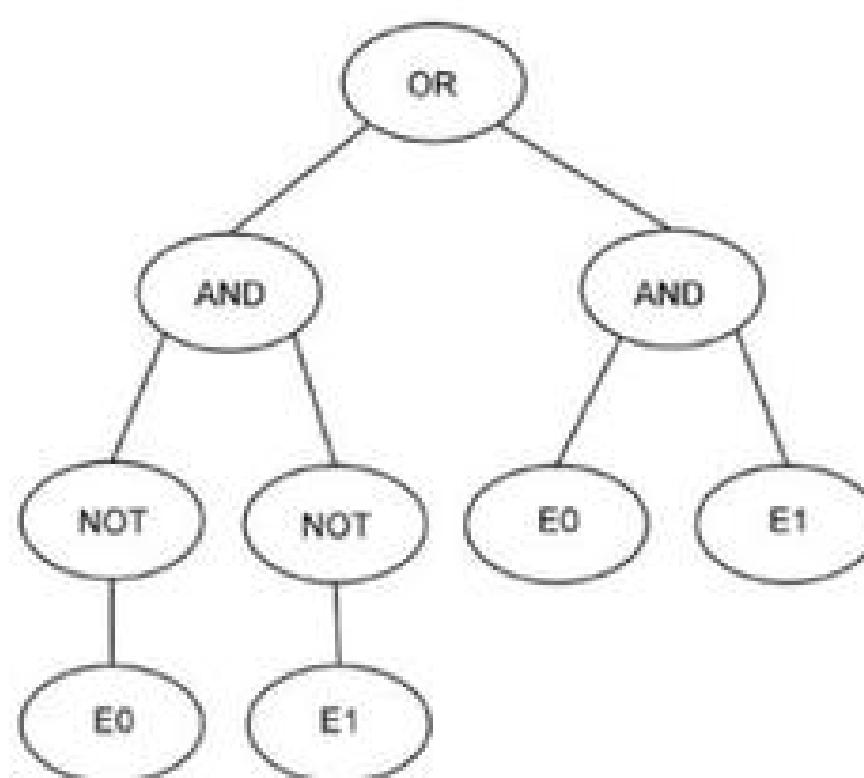
## Representación de las soluciones

**Codificación binaria** En el enfoque más conocido de entre los métodos de computación evolutiva, los *algoritmos genéticos*, se utiliza la codificación binaria. De este modo, cada posible solución se representa mediante una cadena de  $l$  bits de longitud. Así, cuando el espacio de búsqueda contenga variables con valores reales, será necesario discretizar la búsqueda del mismo modo que en el ejemplo anterior.

**Codificación real** Para simplificar, en muchos casos se trabaja con codificaciones más cercanas al problema, como la representación directa de los parámetros reales. La representación de una solución mediante un vector de reales es la técnica habitualmente empleada en las llamadas *estrategias evolutivas*, probablemente porque surgieron en el contexto de aplicaciones a problemas de ingeniería caracterizados por parámetros continuos.

**Codificación de programas** En los problemas en los que no se conoce la forma general que puede adoptar la solución y por tanto no se puede expresar simplemente mediante un conjunto de parámetros es necesario acudir a representaciones más elaboradas, como pueden ser las empleadas en *programación genética*. La idea de este enfoque consiste en representar las soluciones como programas. No obstante, la mayor parte de los lenguajes de programación no son adecuados para este propósito, ya que la combinación de dos programas (por ejemplo en C) o su variación aleatoria no constituirá generalmente un programa sintácticamente válido. Por ello se suele trabajar directamente con programas expresados en forma de árbol (como el árbol sintáctico que generaría un compilador). Un ejemplo de este tipo de representación puede observarse en la Figura 4.42. Alternativamente, se puede trabajar en LISP, ya que en este lenguaje el código fuente es prácticamente equivalente al árbol sintáctico del programa.

Para solucionar un problema empleando programación genética es necesario considerar las funciones y operaciones que se van a emplear (el llamado *conjunto de funciones*), y las variables sobre



**Figura 4.42.** Función binaria de paridad par con dos entradas ( $E_0$  y  $E_1$ ) representada mediante programación genética.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

2. Suponer un modelo de neurona, distinto al utilizado en el perceptrón, que emplea las siguientes funciones de combinación y activación:

$$\begin{aligned} \text{net} &= \sum_i (x_i - w_i)^2 \\ f(\text{net}) &= e^{-\text{net}} \end{aligned}$$

- ¿Qué forma geométrica tienen las regiones de decisión que producen este tipo de neuronas?
  - Desarrollar una regla de actualización de pesos basada en descenso por gradiente para entrenar una red formada por una única neurona de este tipo.
3. El problema de aprender la función de paridad impar (función XOR) está relacionado con los problemas de **suma módulo n**. Por ejemplo, el problema de **suma módulo 2** para dos entradas consistiría en asociar a cada posible par de entradas el valor de su suma módulo 2:

|     |   |     |   |
|-----|---|-----|---|
| 0   | 0 | --> | 0 |
| 0   | 1 | --> | 1 |
| 0   | 2 | --> | 0 |
| ... |   |     |   |
| 1   | 0 | --> | 1 |
| 1   | 1 | --> | 0 |

¿Por qué este tipo de problemas es especialmente difícil de resolver utilizando un perceptrón multicapa?

4. En una memoria asociativa, ¿qué relación guarda la ortogonalidad de los patrones de entrada con la calidad de la recuperación de los mismos? Plantear un ejemplo numérico de dimensión  $N = 2$  en donde los patrones sean ortogonales y comprobar que dado un ejemplo de entrada ligeramente distinto de uno de los patrones almacenados se garantiza la correcta recuperación de los patrones.
5. ¿Qué impacto tiene la hipótesis de independencia estadística en el problema del filtro de mensajes? Escribir la función de verosimilitud suponiendo que la probabilidad de una palabra depende tanto de la anterior como de la posterior. ¿Cómo se traduce esto a nivel de coste espacial y temporal de cara a la evaluación de dicha verosimilitud? Poner un ejemplo de casos en donde este nuevo modelo permitiría discriminar mejor que el modelo que asume independencia.
6. El *problema de la moneda trucada* es un ejemplo sencillo de aplicación de los HMMs y del algoritmo Baum-Welch. Asumiendo que tenemos una moneda que puede estar en dos estados (cara) o (cruz), se trata de determinar empíricamente si dicha moneda está trucada o no. Para ello, se construye el HMM asociado (dos estados y cuatro transiciones) y se trataría de estimar sus pesos para un conjunto de secuencias de observaciones. Supongamos que se observa la secuencia *cxcx*, ¿cuál sería el HMM asociado?, ¿y si se observa *xxcc*? Aplicar el algoritmo de Baum-Welch a cada secuencia y razonar el resultado obtenido.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

## **Parte II**

# **Áreas de aplicación**



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



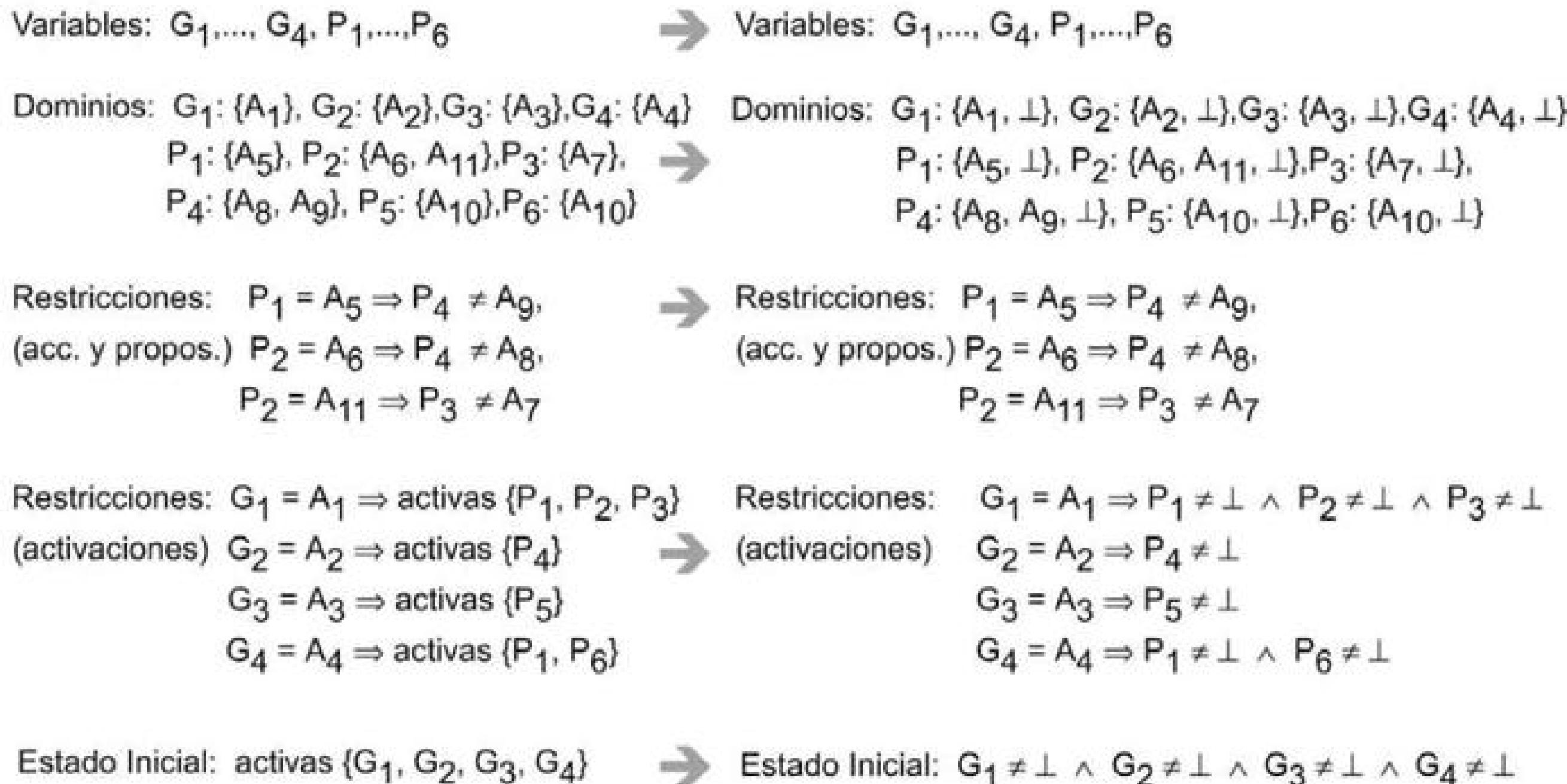
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



**Figura 5.18.** Conversión de un problema DCSP en otro CSP.

convierte en la restricción CSP estándar:  $G_1 = A_1 \Rightarrow P_1 \neq \perp \wedge P_2 \neq \perp \wedge P_3 \neq \perp$ . Por lo tanto la activación de las restricciones tiene que ver únicamente con el hecho de asegurar que las proposiciones que son precondiciones de una acción seleccionada toman valores distintos de  $\perp$ .

Finalmente, una restricción de exclusividad entre dos proposiciones con la forma:

$\neg(\text{activas}(p_{11}) \wedge \text{activas}(p_{12}))$  se traduce como:  $\neg(p_{11}) \neq \perp \wedge \neg(p_{12}) \neq \perp$

Las restricciones de exclusividad entre acciones ya están en formato CSP. Una vez que todas las restricciones se encuentran en formato CSP pueden ser resueltas por cualquier técnica estándar de búsqueda CSP.

Otra alternativa consiste en considerar el problema de planificación como un problema de satisfacibilidad (*SAT problem*). En este caso el problema se codifica como un conjunto de axiomas, basados en lógica proposicional, de forma que se intenta encontrar una asignación de valores a las variables que satisfagan dichos axiomas. Concretamente, se utilizan fórmulas CNF, cuya estructura es:  $C_1 \wedge C_2 \wedge \dots \wedge C_m$ , en donde cada  $C_i$  es una disyunción de la forma  $(z_1 \vee z_2 \vee \dots \vee z_k)$ , siendo cada  $z_j$  un literal de un conjunto de  $(x_1, x_2, \dots, x_n)$  de variables booleanas y sus negaciones, y en donde el número de literales  $k$  en la cláusula varía según  $i$ , con  $i = 1, 2, \dots, m$ . Por lo tanto, el problema es determinar si existe una asignación de valores booleanos a las variables  $x_i$  de forma que la fórmula sea cierta (el dominio de las variables tiene cardinalidad 2).



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

Las medidas de textura se usan como base para tomar decisiones heurísticas. Una medida de textura es una evaluación de propiedades de un grafo de restricciones y refleja la estructura intrínseca de un problema particular. Las primeras medidas de textura que fueron aplicadas a un problema de *scheduling* son las denominadas *contention* y *reliance*, debidas a Sadeh; la primera es usada para identificar el recurso más crítico en un momento determinado, mientras que la segunda identifica las dos actividades que “más” compiten por dicho recurso en ese momento. Otro ejemplo de medida de textura es la del cálculo de la probabilidad de fallo de una variable. Dicha probabilidad se estima teniendo en cuenta el tamaño del dominio de las variables: cuanto más pequeño sea el dominio, más alta será la probabilidad de fallo. Otras medidas de textura son las denominadas *slack* y *bslack*. La primera de ellas calcula la holgura resultante de secuenciar dos actividades en un determinado orden. La segunda es una estimación de la contención entre dos actividades: cuanto más pequeño es el valor de *bslack*, mayor es la contención entre las dos actividades.

## Métodos de búsqueda local

Los métodos de búsqueda local son aquéllos que trabajan sobre una serie de configuraciones definidas, es decir, un conjunto finito de soluciones, una función de coste y un mecanismo de generación que permite generar una transición de una configuración a otra mediante un pequeño cambio. El objetivo es modificar la configuración actual de forma sucesiva para dirigir la búsqueda a una mejor solución.

En la búsqueda local, los mecanismos de configuración determinan un vecino para cada configuración. Un *vecino*,  $N(x)$ , es una función que define una transición simple de una solución  $x$  a otra, induciendo un cambio que puede verse típicamente como una pequeña perturbación. Cada solución puede alcanzarse directamente desde  $x$  a partir de una transformación parcial predefinida de  $x$ , llamada *movimiento* ( $x'$ ), y decimos que  $x$  se mueve a  $x'$  cuando se realiza dicha transición. Normalmente se asume una simetría en la mayoría de búsquedas de vecinos, de forma que  $x$  es vecino de  $x'$  si y sólo si  $x'$  es vecino de  $x$ . La idea es perturbar de forma progresiva la configuración actual a través de una sucesión de vecinos para dirigir la búsqueda hacia una solución mejor. En los casos en los que las soluciones puedan contener inconsistencias, la mejora de la solución se suele definir como relativa a un objetivo modificado que penaliza dichas inconsistencias. En cualquier caso, la selección de un vecino es dictada por algún criterio extra (seleccionar el vecino con menor coste, seleccionar el mejor de entre un conjunto, etc.).

Desde una perspectiva general, la solución a un problema de *scheduling* puede considerarse como una colección de decisiones locales concernientes a qué operación se debe secuenciar en siguiente lugar.

Uno de los grupos más populares de métodos de búsqueda iterativa son los llamados *algoritmos umbral*, que eligen una nueva configuración si la diferencia de coste entre la solución actual y su vecina está por debajo de un umbral dado. Dentro de éstos es conocida la técnica de *enfriamiento simulado* (*Simulated Annealing*), en la que los umbrales son positivos y estocásticos. El enfriamiento simulado es una técnica de búsqueda aleatoria que fue introducida a partir de la física estadística de la simulación por computador de un proceso de enfriamiento de un metal caliente, hasta que éste alcanzó su estado de mínima energía. En este tipo de métodos, las configuraciones son análogas a los estados de un sólido, utilizando una función de coste y un parámetro de control equivalentes a la energía y la temperatura, respectivamente.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



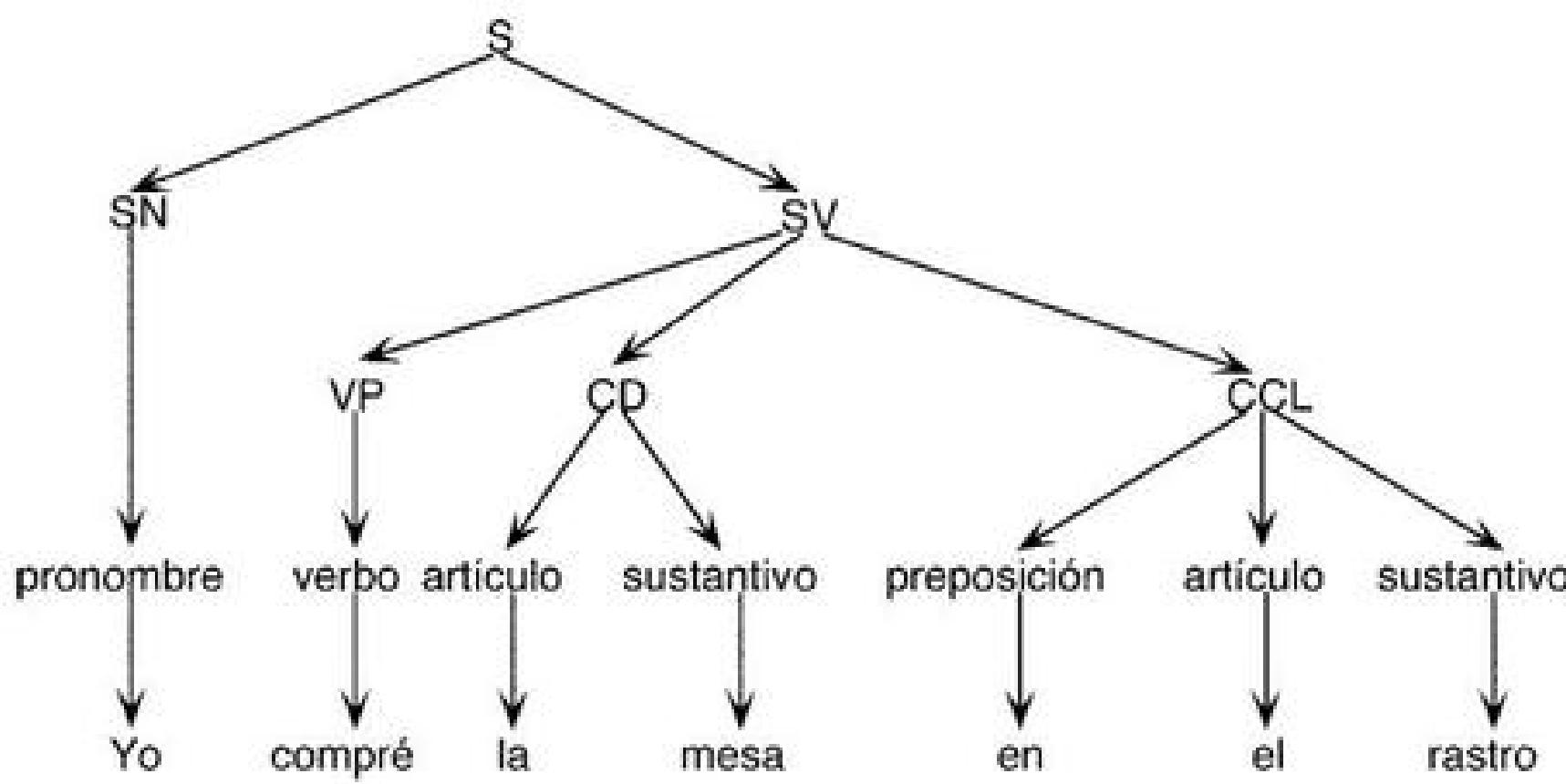
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



**Figura 6.2.** Ejemplo de árbol sintáctico.

Una vez analizada morfológica y sintácticamente la frase, debemos dar significado a partir de un diccionario semántico. Dicho diccionario puede tener una estructura similar a la siguiente:

```

(querer (es evento-mental)
 (agente persona)
 (objeto evento-fisico)
)
(imprimir (es evento-fisico)
 (agente persona o programa)
 (objeto fichero o salida-programa)
)
(usuario (es persona)
 (identificador a00000)
 (nombre Juan Garcia)
)
(usuario (es persona)
 (identificador a00001)
 (nombre Miguel)
)
(objeto (es fichero)
 (identificador lp0000)
 (propietario a00000)
 (nombre .profile)
)
)

```

Por supuesto dicho diccionario deberá ser mucho más extenso y se actualizará con cada acción ejecutada (borrado de ficheros, creación de ficheros nuevos, etc.). El análisis semántico de la frase anterior deberá proporcionar una salida del estilo:



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

estructuras sintácticas a la forma lógica vamos a llamarlo *interpretación semántica*. Por otro lado, al proceso de integrar el contexto para producir la representación final en la base de conocimiento, a partir de la forma lógica, se le llamará *interpretación contextual*. El proceso al completo se puede observar en la Figura 6.18.

No es objetivo de estos temas el proponer mecanismos de representación del conocimiento para el lenguaje natural. Vamos a suponer que el lenguaje de representación que utilizamos es el del cálculo de predicados de primer orden.

## Forma lógica

Para empezar a procesar semánticamente una frase debemos definir las primitivas básicas de procesamiento. Vamos, pues, a definir un lenguaje en el cual podamos combinar estos elementos para formar significados de expresiones más complejas. La primitiva básica que vamos a manejar en el análisis semántico es el significado de las palabras. Este significado servirá como átomo de la representación. Los distintos tipos de átomos pueden clasificarse por los tipos de elementos que describen:

- **Términos.** Describen objetos del mundo (incluyendo objetos abstractos, tales como eventos y situaciones).
- **Predicados.** Describen relaciones y propiedades.

La forma lógica de una frase se forma con un predicado seguido de un número apropiado de términos (como si fueran sus argumentos). Ejemplos de predicados son los siguientes:

| Frase                | Proposición            |
|----------------------|------------------------|
| “Juan es inglés”     | (INGLES1 JUAN1)        |
| “Juan odia a Amparo” | (ODIAR1 JUAN1 AMPARO1) |

En la primera frase estamos utilizando un predicado monario (también llamados *propiedades*). El predicado monario se forma a partir de nombres comunes, tales como *inglés*, *perro*, etc. Los nombres propios forman términos (*Juan*). Los verbos, en general, serán predicados de *n* argumentos, dependiendo del número de términos que aparezca. Por otro lado tenemos una nueva clase de átomos que se denominan *operadores lógicos* y que sirven para construir proposiciones más complejas. Un ejemplo puede ser el operador lógico *NO*, que permite negar una proposición:

“Juan no odia a Amparo” (NO (ODIAR1 JUAN1 AMPARO1))

**Operadores lógicos** Otros tipos de operadores lógicos son los ya conocidos: Y, O, implica, etc. Con estos átomos únicamente podemos definir un conjunto muy restringido de sentencias posibles. Debemos definir constructores semánticos adicionales para poder abordar frases más complejas. Estos nuevos operadores hacen referencia a términos en el lenguaje natural, tales como: *todos*, *algunos*, *muchos*, *pocos*, *el*, etc. Vamos a definir estos operadores como “cuantificadores lógicos”. En el cálculo de proposiciones de primer orden existen dos cuantificadores:  $\forall$  y  $\exists$ . En lenguaje natural, además, debemos emplear los antes descritos. Vamos a emplear estos cuantificadores de la siguiente manera:



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

"Estoy esperando que me llamen por teléfono."

Al procesarla se nos plantea una ambigüedad al tener dos posibles significados:

- En este preciso momento estoy al lado de un teléfono y espero una llamada urgente. Se limita la capacidad de acción del hablante.
- En estos días espero una llamada, posible contestación de una entrevista de trabajo. Aquí, la libertad de acción del hablante es superior a la anterior interpretación.

Debemos, pues, mantener información del contexto en el que estamos hablando. Cuando analizamos una frase debemos pensar que incorpora estructuras, tanto sintácticas como semánticas, que pueden ser antecedente de las frases siguientes o que pueden haber sido mencionadas en frases anteriores. Un ejemplo más concreto es el siguiente:

"Juan está buscando su perro."

"Él piensa que lo encontrará."

El pronombre *él* hace referencia a *Juan* y *lo* a *su perro*, por lo que al analizar la primera frase debemos almacenar la información correspondiente y al analizar la segunda debemos eliminar la ambigüedad del significado de los pronombres para identificarlos con las entidades de la primera frase. Pero este problema no surge únicamente en los pronombres. En frases del estilo de:

"Paco perdió su perro."

"Juan también."

estamos haciendo referencia a una acción, no únicamente a cosas o personas.

**Lista de entidades del discurso** Con estos ejemplos hemos querido mostrar la necesidad de realizar una integración del discurso. Para realizar esta integración haremos uso de una *lista de entidades del discurso*. Esta lista estará compuesta por una serie de entidades que están definidas en la base del conocimiento y que se han referenciado en las últimas frases. Debemos, pues, identificar los objetos o conceptos referenciados en las frases con los de nuestra base de conocimiento. Estas entidades servirán para eliminar la ambigüedad de las frases tales como los ejemplos antes mencionados. La razón de mantener esta lista de entidades (no únicamente de la última frase) es crear un foco de atención (muy utilizado en percepción). El foco de atención nos permite, de forma cercana a como lo hacemos los humanos, centrarnos en un tema y potenciarlo para que sirva de base para la eliminación de la ambigüedad. Por otro lado, el orden al analizar las frases puede cambiar el significado de éstas. Veamos un ejemplo donde el orden de las frases altera el significado de éstas:

**Discurso 1**

Fui a Madrid.

Compré un regalo.

Volví al día siguiente.

Lo envié por correo.

**Discurso 2**

Fui a Madrid.

Volví al día siguiente.

Compré un regalo.

Lo envié por correo.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



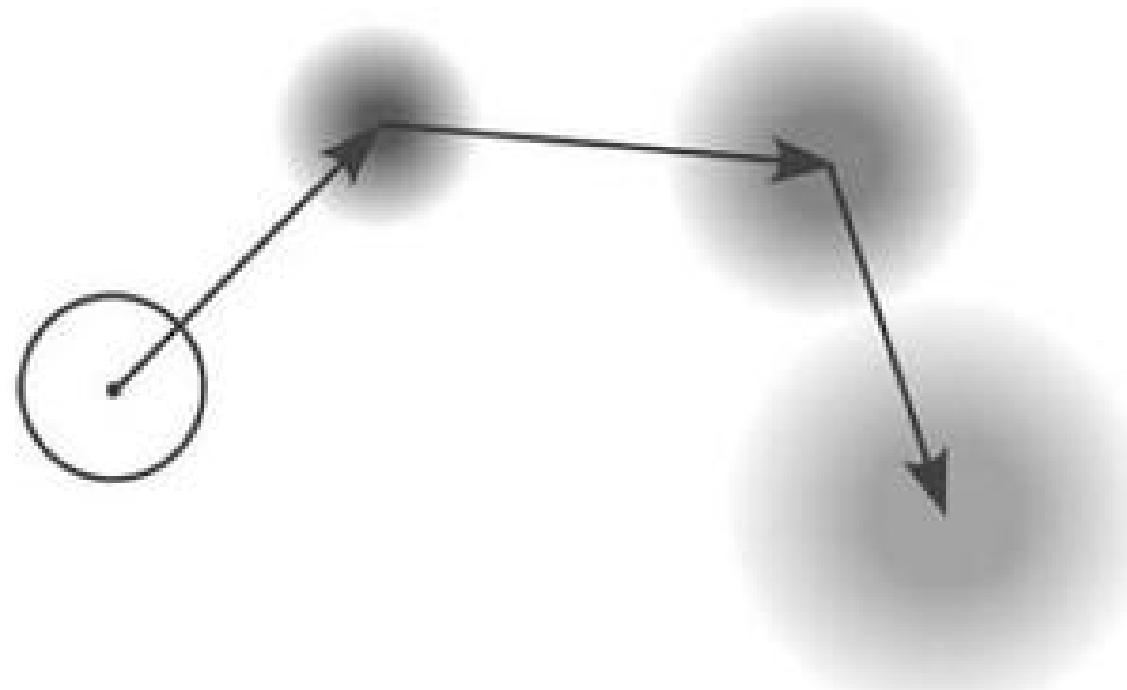
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

asociado. Además, el error es acumulativo, puesto que en un desplazamiento el robot tiene un cierto error asociado (sabemos que está en una posición con una cierta probabilidad, pero también es posible que se encuentre en otra con una probabilidad menor) y en el siguiente movimiento el error aumenta (véase Figura 8.6).

---



**Figura 8.6.** Error producido al desplazarse el robot. La zona sombreada indica la probabilidad de encontrarse en esa posición. Como podemos observar, a cada movimiento la incertidumbre aumenta.

---

**Errores en odometría** Los errores producidos son de dos tipos: sistemáticos y no sistemáticos. Los sistemáticos son los que se producen siempre y requieren una calibración para su eliminación. Un posible error sistemático es la diferencia en el tamaño de las ruedas, que puede provocar que si tenemos una conducción diferencial el robot no pueda ir en línea recta. Los errores no sistemáticos se producen por elementos externos como la superficie de desplazamiento (rugosa, resbaladiza, etc.). Para eliminar este tipo de error se puede intentar modelarlo o utilizar otro tipo de sensor adicional.

## Sensores de localización

**Sistema de posicionamiento global** Un sistema sensorial ampliamente utilizado en robótica para resolver los problemas de localización es el Sistema de Posicionamiento Global o GPS (*Global Positioning System*). Este sistema utiliza un equipo receptor (véase la Figura 8.7) que recibe señales de satélites que se encuentran orbitando alrededor de la tierra. Este receptor necesita, al menos, cuatro satélites visibles para, mediante triangulación, proporcionar la latitud, longitud y altitud del equipo. Este tipo de sistema es ampliamente utilizado en navegación comercial (aviones, barcos, camiones, etc.) y aplicaciones militares (guiado de misiles). Hasta principios de 2000, el ejército de Estados Unidos introducía un error en los satélites (de más de 100 m) para que ninguna fuerza enemiga pudiera hacer uso del sistema. Ahora el error está acotado aproximadamente entre 10 y 15 metros. Sin embargo, se han desarrollado sistemas que reducen el error a menos de 1 metro. El sistema se conoce como DGPS (GPS diferencial) y consiste en utilizar dos receptores, uno fijo y el otro en el robot. Los dos reciben la misma señal, pero si el fijo detecta algún cambio en su posición, este cambio es debido al error y puede ser eliminado del receptor en el robot. El precio de un receptor GPS ronda los 300 euros, mientras que el sistema diferencial puede llegar a los 30.000 euros.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

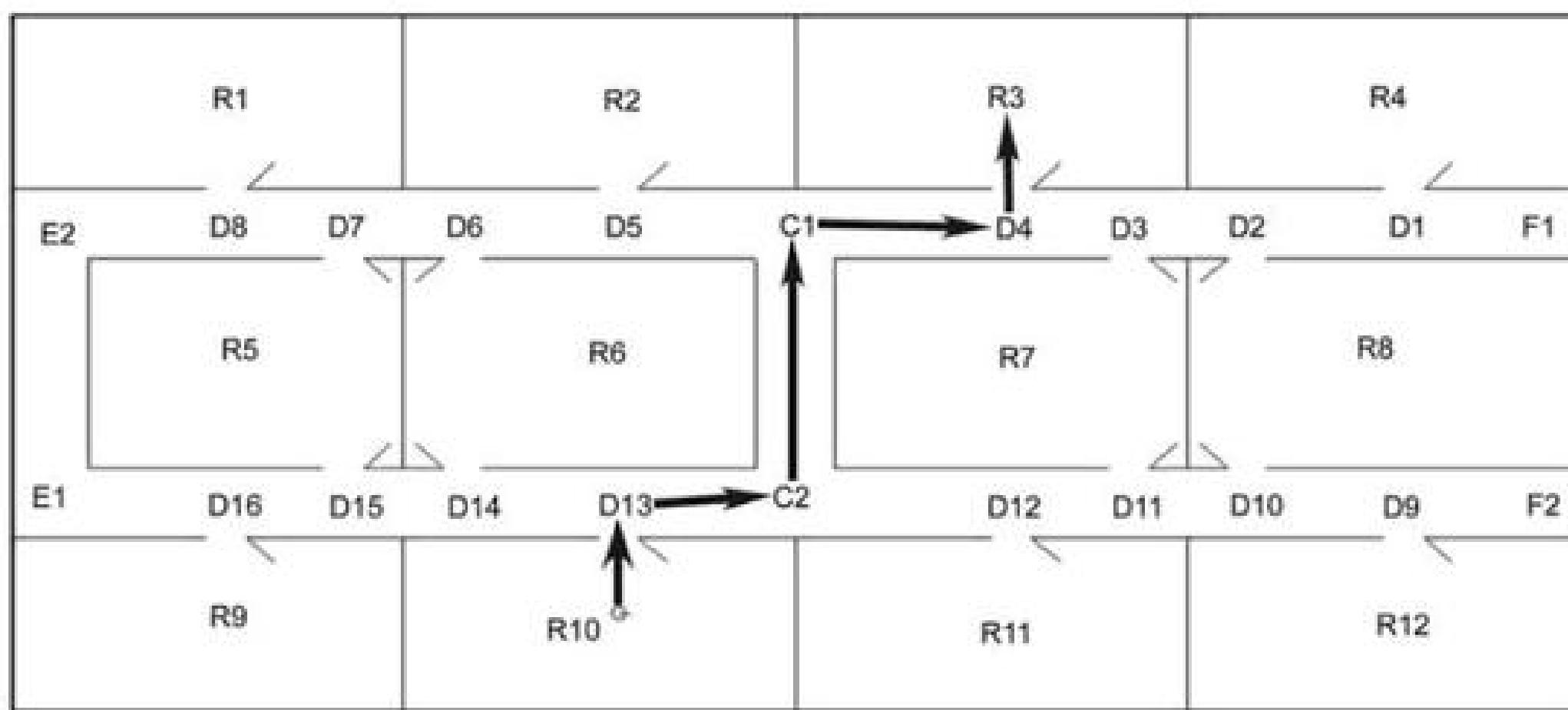


Figura 8.43. Camino encontrado para el punto de partida R10 hasta R3.

cruzarla. En este caso debemos orientar el robot en dirección norte, buscar una puerta y atravesarla. En el siguiente paso, de D13 a C2, debemos orientar el robot hacia el este, seguir el pasillo y detectar que hemos encontrado un cruce. Haciendo uso de esta tabla y del camino obtenido, procedemos a recorrer el camino pasando de un punto distintivo a otro. A continuación se muestra una parte de la salida del proceso, que se encarga de guiar el robot por este camino:

Realizando R10 a D13

Girando al robot hacia el Norte.

Buscando puerta.

Puerta encontrada.

Ejecutando Cruzar-puerta.

Puerta cruzada.

Realizando D13 a C2

Girando al robot hacia el Este.

Ejecutando Seguir-pasillo.

Buscando cruce.

Cruce encontrado. Paro Seguir-Pasillo.

Realizando C2 a C1

Girando al robot hacia el Norte.

Ejecutando Seguir-pasillo.

Buscando cruce.

Cruce encontrado. Paro Seguir-Pasillo.

**Obstáculos no modelados** ¿Qué ocurre cuando nos encontramos con algún obstáculo no modelado? Por ejemplo, un pasillo se encuentra bloqueado. Podemos realizar una nueva búsqueda en el grafo cambiando las aristas que se vean afectadas. En el caso de la Figura 8.44, vemos que el pasillo entre



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

# Índice alfabético

- Árbol de decisión, 136
- Árboles de búsqueda en juegos, 30
- Área USAN, 256
- Admisibilidad  $\varepsilon$ , 27
- Admisibilidad de  $A^*$ , 23
- Aibo, 357
- Algoritmo
  - $A^*$ , 21
  - $A_\varepsilon^*$ , 27
  - AC1, 50
  - AC3, 50
  - $\alpha - \beta$ , 33
  - $\alpha - \beta$  con memoria, 38
  - de Metropolis, 12
  - IDA*<sup>\*</sup>, 28
  - MTD-f, 37
- Algoritmo de Baum-Welch, 130
- Algoritmo de retropropagación, 95
- Algoritmo EM, 131
- Ambigüedad, 205
- Anisotropía, 282
- Arco activo, 216
- Arquitecturas de conductas, 311
- Atractor, 113
- Autómata probabilístico, 124
- Backjumping*, 44, 47
- Backtracking*, 44
- Balizas activas, 315
- Banco de filtros, 269
- Blobs, algoritmo de, 294
- Cálculo lambda, 230
- Canny
  - hysteresis*, 253
  - supresión, 253
  - trade-off*, 254
- Capa oculta, 95
- Capacidad de almacenamiento, 114
- CIE-Lab, modelo, 274
- Cláusulas, 59
  - de Horn, 67
- CLIPS, 72–74
  - hechos, 72
- inferencia, 74
- reglas, 72
- Clustering*, 114
- Clustering EM*, 284
  - inicialización, 285
  - paso E, 285
  - paso M, 285
- Coeficiente de proporcionalidad, 95
- Composición máx-mín, 81
- Computación evolutiva, 140
- Conflictos, 47
- Conjunto *IRIS*, 118
- Conjunto de entrenamiento, 105
- Conjuntos fuzzy, 75
  - operaciones con, 79
- Consistencia
  - global, 53
  - local, 53
- Consistencia de arista, 47
- Convolución, 246
- Cornerness*, 255
- Corners*, 254
- Cortes  $\alpha$  y  $\beta$ , 33
- Coste de corte, 29
- Crisp*, 75
- Cruce, 142, 145
  - en un punto, 148
  - n-puntual, 148
  - uniforme, 148
- CSP, 41
- Deconvolución, 247
- Definición de robot, 307
- Defuzzyfication*, 85
- Descenso por gradiente, 103
- Detector
  - de Beaudet, 256
  - de Canny, 251
  - de Moravec, 255
  - de Nitzberg/Harris, 257
  - de SUSAN, 256
- Difusión anisotrópica, 282
  - enfoque robusto, 284
  - imágenes en color, 284
- Dispositivos CCD, 245

- Distancia de Hamming, 109  
Distribución de Gibbs, 10  
Dominios CSP, 41  
*Edges*, 250  
Encadenamiento  
    hacia adelante, 70  
    algoritmo, 70  
    hacia atrás, 69, 70  
    algoritmo, 70  
    hacia delante, 69  
Entrada neta, 92  
Entrenamiento, 103  
Entrenamiento *on-line*, 103  
Entropía, 137  
Epochs, 100  
Error de cuantificación, 117  
Error de patrón, 100  
Error topológico, 117  
Esqueleto, 263  
Esquema asíncrono, 110  
Esquema geométrico, 17  
Esquema logarítmico, 17  
Estructurado, enfoque, 294  
Fases en el procesamiento del lenguaje, 201  
Filtrado no-lineal, 107  
Fisher, test de, 270  
Forma normal, 59  
    conversión a, 62  
Forma, descriptor de, 261  
Función de adecuación, 142  
Función de energía, 10  
función de evaluación, 19  
Función de Lyapunov, 111  
Función de pertenencia, 75  
Función escalón, 93  
Funciones de activación, 96  
*Fuzzificación*, 85  
Gabor, filtros de, 269  
Ganancia de información, 137  
Geometría epipolar, 321  
GPS, 314  
Grafo de regiones, 294  
Grafo de visibilidad, 334  
Grafos de restricciones, 43  
Gramáticas independientes del contexto, 209  
Grupo de secuencias base, 124  
Heurística, 17  
admisible, 24  
focal, 27  
Histograma de textura, 270  
Holonómico, 329  
Horizonte de juego, 30  
HSB, modelo, 273  
IA, 3  
Icónico, enfoque, 289  
ID3, algoritmo, 137–140  
Implicación difusa, 80  
Inteligencia Artificial, 3  
Kolgomorov-Smirnov, test de, 271  
Kullback-Leibler, distancia de, 272  
Landmarks, 340  
Máximo a posteriori y máxima verosimilitud, 121  
Métodos de ventana nula, 36  
Mínima energía, 11  
Mínimo global, 15  
Mínimo local, 14  
Mínimos locales, 338  
Mapas auto-organizativos, 114  
*Matching* de grafos, 296  
Matriz de *matching*, 296  
Matriz de pesos, 106  
Memoria auto-asociativa de Hopfield, 109  
Minerva, 356  
Minimax, 30  
Minimización de energía, 111  
Minkowski, 330  
Modelo de ADN, 123  
Modelos ocultos de Markov, 124  
Modelos ocultos de Markov (HMMs), 123  
Modificadores lingüísticos, 77  
*modus ponens* difuso, 81  
Momento, 103  
Momento o componente inercial, 103  
Momentos, 261  
Mutación, 142, 145  
Neurona artificial, 92  
Neurona ganadora, 114  
Neuronal  
    umbral, 94  
Neuronas  
    pesos de, 94  
Neuronas biológicas, 91  
Neuronas e hiperplanos, 93

- Neuronas frontera, 118  
 Nivel de información, 25  
 No-separabilidad lineal, 95  
 No-supervisado, 114  
 NP-completitud, 9
- Odometría, 313  
 Operador  
     de Sobel, 251  
     gradiente, 251  
 Orden de selección  
     de valores, 43  
     de variables, 43  
 Organización cortical, 114  
*Overshooting*, 103
- Paradigma conexionista, 92  
 Paradigma jerárquico, 310  
 Paradigma reactivo, 311  
*Pathfinding*, 17  
 PCA, 135, 290  
     aprendizaje, 290  
     reconocimiento, 293  
 Perceptrones  
     fase *backward*, 99  
     fase *forward*, 99  
 Perceptrones multi-capa, 95  
 Plasticidad neuronal, 92  
 Poda  $\alpha - \beta$ , 32  
*Point-spread function*, 246  
 Preservación de aristas, función de, 283  
 Probabilidad *a priori*, 120  
 Probabilidades de observación, 125  
 Probabilidades de transición, 124  
 Probabilidades iniciales, 124  
 Problema de la correspondencia, 320  
 Problema de la evaluación, 125  
 Problema del viajante de comercio, 9  
 Problemas de clasificación, 95  
 Problemas de paridad de la XOR, 95  
 Problemas NP-completos, 9  
 Propagación de restricciones, 47  
 Programación genética, 147  
 Programación lógica, 67  
 PROLOG, 67–70  
 Puntos distintivos, 340
- Radio de vecindad, 115  
 Re-estimación de  $\lambda$ , 127  
 Reconocimiento de palabras aisladas, 134
- Redes neuronales, 91  
 Reemplazo, 143, 145  
 Regla de actualización de pesos, 114  
 Regla de la cadena, 101  
 Regla delta +, 94  
 Reglas, 69  
 Reglas de morfología, 207  
 Resolución, 59  
     algoritmo, 59  
     algoritmo proposicional, 60  
     en lógica de primer orden, 65  
     lineal, 68  
     regla de, 59  
 Restricciones  
     binarias, 41  
     unarias, 41  
 Restricciones CSP, 41  
 RGB, modelo, 273  
 Robota, 308  
 Rol temático, 231  
 Ruido de recuperación, 106
- Satisfacción de restricciones, 41  
 Segmentación, 277  
 Selección, 142, 144  
     proporcional, 150  
     torneo, 150  
 Sensor, 313  
 Sensor de ultrasonidos, 316  
 Sensor láser, 317  
 Sigmoide, 96  
*Simulated annealing*, 10  
*Singleton*, 77, 82  
 Sistema dinámico, 111  
 Sistemas expertos, 69  
     difusos, 84  
 Skolem, función de, 63  
*Snakes*, 277  
     energía de continuidad, 278  
     energía de curvatura, 279  
     energía externa, 278  
     función de energía, 279  
     método del balón, 280  
     programación dinámica, 280  
*Snaxels*, 278  
*Soft Assign*, algoritmo, 297  
 Solución factible, 41  
 SOMs, 114  
 Suma de correlaciones, 107

- Test de Turing, 5
- Test no paramétrico
  - $\chi^2$ , 271
  - Kolgomorov-Smirnov, 271
- Textura, modelo de, 268
- Transformada de ejes medios, 263
- Transformada de Hough, 258
  - círculo, 260
  - recta, 258
- TSP, 9
- Unificación, 64, 70
  - algoritmo, 65
- Vértice del hipercubo, 112
- Valor minimax del nodo raíz, 32
- Variabilidad, factores de, 290
- Variable  $\alpha_t(i)$ , 125
- Variables  $\gamma_t(i)$ , 127
- Variables  $\xi_t(i, j)$ , 128
- Variables *backward*, 126
- Variables *forward*, 126
- Variables CSP, 41
- Variables lingüísticas, 79
- Vecindad, 263
- Verosimilitud, 120
- Visión estéreo, 320

# Inteligencia Artificial

Modelos, Técnicas  
y Áreas de Aplicación

**¿QUÉ ES INTELIGENCIA ARTIFICIAL (IA)?** Esta disciplina, o podríamos decir *inter-disciplina* (confluencia de la lógica matemática, ciencia cognitiva, computación, automática, filosofía de la ciencia...) viene ocupándose, desde hace casi medio siglo, de la **programación de computadores, robots, dispositivos, agentes software, sistemas expertos y tutores**, etc., para resolver problemas, es decir, para desarrollar habilidades, que de una manera u otra exigen cierto grado de inteligencia en los humanos.

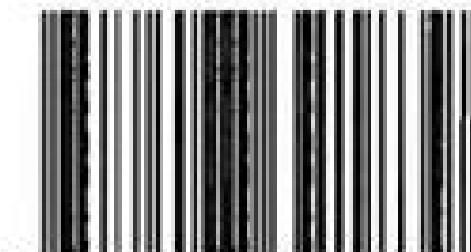
Algunos de estos problemas, como **Jugar al ajedrez**, difíciles para nosotros por exigir múltiples niveles de razonamiento, han resultado ser alcanzables para las máquinas (en mayo de 1997 Deep Blue, un supercomputador masivamente paralelo, derrotó a Gary Kasparov). Sin embargo, el juego del ajedrez es uno de los pocos dominios en donde la IA ha satisfecho sus expectativas iniciales. Un ejemplo de lo contrario es la **visión artificial**. A pesar de que somos capaces de interpretar sin esfuerzo aparente lo que vemos, no existe hasta la fecha un sistema computacional con capacidades visuales cercanas a las humanas. La interpretación del **lenguaje natural** con todos sus matices y la **traducción automática** entre idiomas, son objetivos aún lejanos. En **robótica**, se están desarrollando robots "relativamente autónomos" en el sentido de que son capaces de navegar de forma segura por el entorno, e incluso de reconstruir desde la nada, complejos mapas del entorno. Pero, aún queda mucho por avanzar en cuanto a la flexibilidad de estos sistemas en entornos cambiantes.

Este libro aborda de forma sistemática los distintos aspectos a cubrir en el apasionante reto de dotar de **Inteligencia a las máquinas**. Estos aspectos, ya asentados en el campo científico de la IA, comprenden: la **búsqueda inteligente de soluciones** entre un elevadísimo número de posibilidades, el **razonamiento automático** preciso y aproximado, el **aprendizaje automático** a partir de ejemplos, el **desarrollo automático** de planes de acción, las técnicas básicas de **interpretación del lenguaje** y de **visión artificial** y la incorporación de estos y otros elementos a sistemas **robóticos autónomos**.

THOMSON

[www.paraninfo.es](http://www.paraninfo.es)  
[www.thomsonlearning.com](http://www.thomsonlearning.com)

ISBN 84-9732-183-9



9 788497 321839