- Por lo general, no es factible considerar el árbol entero de juegos (hasta con alfabeta), entonces tenemos que cortar la búsqueda en algún punto y aplicar una **función de evaluación** que dé una estimación de la utilidad de un estado.
- Los juegos de azar pueden manejarse con una extensión del algoritmo minimax que evalúa un **nodo de posibilidad** tomando la utilidad media de todos sus nodos hijos, ponderados por la probabilidad de cada hijo.
- El juego óptimo en juegos de **información imperfecta**, como el bridge, requiere el razonamiento sobre los **estados de creencia** actuales y futuros de cada jugador. Una aproximación simple puede obtenerse haciendo un promedio del valor de una acción sobre cada configuración posible de la información ausente.
- Los programas pueden equipararse o pueden ganar a los mejores jugadores humanos en damas, Otelo y *backgammon,* y están cercanos en bridge. Un programa ha ganado al campeón mundial de ajedrez en un partido de exhibición. Los programas permanecen en el nivel aficionado en Go.

Notas bibliográficas e históricas

La temprana historia de los juegos mecánicos se estropeó por numerosos fraudes. El más célebre de estos fue «El Turco» de Baron Wolfgang von Kempelen (1734-1804), un supuesto autómata que jugaba al ajedrez, que derrotó a Napoleón antes de ser expuesto como la caja de bromas de un mago que escondía a un humano experto en ajedrez (*véase* Levitt, 2000). Jugó desde 1769 hasta 1854. En 1846, Charles Babbage (quien había sido fascinado por el Turco) parece haber contribuido a la primera discusión seria de la viabilidad del computador de ajedrez y de damas (Morrison y Morrison, 1961). Él también diseñó, pero no construyó, una máquina con destino especial para jugar a tic-tac-toe. La primera máquina real de juegos fue construida alrededor de 1890 por el ingeniero español Leonardo Torres y Quevedo. Se especializó en el «RTR» (rey y torre contra el rey), la fase final de ajedrez, garantizando un triunfo con el rey y torre desde cualquier posición.

El algoritmo minimax se remonta a un trabajo publicado en 1912 de Ernst Zermelo, el que desarrolló la teoría moderna de conjuntos. El trabajo, lamentablemente, tenía varios errores y no describió minimax correctamente. Un fundamento sólido, para la teoría de juegos, fue desarrollado en el trabajo seminal de *Theory of Games and Economic Behaviour* (von Neumann y Morgenstern, 1944), que incluyó un análisis en el que mostraba que algunos juegos *requieren* estrategias aleatorizadas (o imprevisibles). *Véase* el Capítulo 17 para más información.

Muchas figuras influyentes de los comienzos de los computadores, quedaron intrigadas por la posibilidad de jugar al ajedrez con un computador. Konrad Zuse (1945), la primera persona que diseñó un computador programable, desarrolló ideas bastante detalladas sobre cómo se podría hacer esto. El libro influyente de Norbert Wiener (1948), *Cybernetics,* habló de un diseño posible para un programa de ajedrez, incluso las ideas de búsqueda minimax, límites de profundidad, y funciones de evaluación. Claude Shannon (1950) presentó los principios básicos de programas modernos de juegos con mucho más detalle que Wiener. Él introdujo la idea de la búsqueda de estabilidad y describió algunas ideas para la búsqueda del árbol de juegos selectiva (no exhaustiva). Slater

(1950) y los que comentaron su artículo también exploraron las posibilidades para el juego de ajedrez por computador. En particular, I. J. Good (1950) desarrolló la noción de estabilidad independientemente de Shannon.

En 1951, Alan Turing escribió el primer programa de computador capaz de jugar un juego completo de ajedrez (*véase* Turing *et al*., 1953). Pero el programa de Turing nunca se ejecutó sobre un computador; fue probado por simulación a mano contra un jugador muy débil humano, que lo derrotó. Mientras tanto D. G. Prinz (1952) escribió , y realmente ejecutó, un programa que resolvió problemas de ajedrez, aunque no jugara un juego completo. Alex Bernstein escribió el primer programa para jugar un juego completo de ajedrez estándar (Bernstein y Roberts, 1958; Bernstein *et al*., 1958)[3].

John McCarthy concibió la idea de la búsqueda alfa-beta en 1956, aunque él no lo publicara. El programa NSS de ajedrez (Newell *et al*., 1958) usó una versión simplificada de alfa-beta; y fue el primer programa de ajedrez en hacerlo así. Según Nilsson (1971), el programa de damas de Arthur Samuel (Samuel, 1959, 1967) también usó alfa-beta, aunque Samuel no lo mencionara en los informes publicados sobre el sistema. Los trabajos que describen alfa-beta fueron publicados a principios de 1960 (Hart y Edwards, 1961; Brudno, 1963; Slagle, 1963b). Una implementación completa de alfa-beta está descrita por Slagle y Dixon (1969) en un programa para juegos de Kalah. Alfa-beta fue también utilizada por el programa «Kotok-McCarthy» de ajedrez escrito por un estudiante de John McCarthy (Kotok, 1962). Knuth y Moore (1975) proporcionan una historia de alfa-beta, junto con una demostración de su exactitud y un análisis de complejidad en tiempo. Su análisis de alfa-beta con un orden de sucesores aleatorio mostró una complejidad asintótica de $O((b/\log b)^d)$, que pareció bastante triste porque el factor de ramificación efectivo $b/\log b$ no es mucho menor que b. Ellos, entonces, se dieron cuenta que la fórmula asintótica es exacta sólo para $b > 1000$ más o menos, mientras que a menudo se aplica un $O(b^{3d/4})$ a la variedad de factores de ramificación encontrados en los juegos actuales. Pearl (1982b) muestra que alfa-beta es asintóticamente óptima entre todos los algoritmos de búsqueda de árbol de juegos de profundidad fija.

El primer partido de ajedrez de computador presentó al programa Kotok-McCarthy y al programa «ITEP» escrito a mediados de 1960 en el Instituto de Moscú de Física Teórica y Experimental (Adelson-Velsky *et al*., 1970). Este partido intercontinental fue jugado por telégrafo. Se terminó con una victoria 3-1 para el programa ITEP en 1967. El primer programa de ajedrez que compitió con éxito con la gente fue MacHack 6 (Greenblatt *et al*., 1967). Su grado de aproximadamente 1.400 estaba bien sobre el nivel de principiante de 1.000, pero era bajo comparado con el grado 2.800 o más que habría sido necesario para satisfacer la predicción de 1957 de Herb Simon de que un programa de computador sería el campeón mundial de ajedrez en el plazo de 10 años (Simon y Newell, 1958).

Comenzando con el primer Campeonato Norteamericano ACM de Ajedrez de computador en 1970, el concurso entre programas de ajedrez se hizo serio. Los programas a principios de 1970 se hicieron sumamente complicados, con varias clases de trucos para eliminar algunas ramas de búsqueda, para generar movimientos plausibles, etcétera.

[3] Newell *et al*. (1958) mencionan un programa ruso, BESM, que puede haber precedido al programa de Bernstein.

En 1974, el primer Campeonato Mundial de Ajedrez de computador fue celebrado en Estocolmo y ganado por Kaissa (Adelson-Velsky *et al*, 1975), otro programa de ITEP. Kaissa utilizó la aproximación mucho más directa de la búsqueda alfa-beta exhaustiva combinada con la búsqueda de estabilidad. El dominio de esta aproximación fue confirmado por la victoria convincente de CHESS 4.6 en el Campeonato Mundial de 1977 de Ajedrez de computador. CHESS 4.6 examinó hasta 400.000 posiciones por movimiento y tenía un grado de 1.900.

Una versión posterior de MacHack, de Greenblatt 6, fue el primer programa de ajedrez ejecutado sobre un *hardware* de encargo diseñado expresamente para el ajedrez (Moussouris *et al*., 1979), pero el primer programa en conseguir éxito notable por el uso del *hardware* de encargo fue Belle (Condon y Thompson, 1982). El *hardware* de generación de movimientos y de la evaluación de la posición de Belle, le permitieó explorar varios millones de posiciones por movimiento. Belle consiguió un grado de 2.250, y se hizo el primer programa de nivel maestro. El sistema HITECH, también un computador con propósito especial, fue diseñado por el antiguo Campeón de Ajedrez de Correspondencia Mundial Hans Berliner y su estudiante Carl Ebeling en CMU para permitir el cálculo rápido de la función de evaluación (Ebeling, 1987; Berliner y Ebeling, 1989). Generando aproximadamente 10 millones de posiciones por movimiento, HITECH se hizo el campeón norteamericano de computador en 1985 y fue el primer programa en derrotar a un gran maestro humano en 1987. Deep Thought, que fue también desarrollado en CMU, fue más lejos en la dirección de la velocidad pura de búsqueda (Hsu *et al*., 1990). Consiguió un grado de 2.551 y fue el precursor de Deep Blue. El Premio de Fredkin, establecido en 1980, ofreció 5.000 dólares al primer programa en conseguir un grado de maestro, 10.000 dólares al primer programa en conseguir un grado FEUA (Federación de los Estados Unidos de Ajedrez) de 2.500 (cerca del nivel de gran maestro), y 100.000 dólares para el primer programa en derrotar al campeón humano mundial. El premio de 5.000 dólares fue reclamado por Belle en 1983, el premio de 10.000 dólares por Deep Thought en 1989, y el premio de 100.000 dólares por Deep Blue por su victoria sobre Garry Kasparov en 1997. Es importante recordar que el éxito de Deep Blue fue debido a mejoras algorítmicas y de *hardware* (Hsu, 1999; Campbell *et al*., 2002). Las técnicas como la heurística de movimiento nulo (Beal, 1990) han conducido a programas que son completamente selectivos en sus búsquedas. Los tres últimos Campeonatos Mundiales de Ajedrez de computador en 1992, 1995 y 1999 fueron ganados por programas que se ejecutan sobre computadores personales. Probablemente la mayor parte de la descripción completa de un programa moderno de ajedrez la proporciona Ernst Heinz (2000), cuyo programa DARKTHOUGHT fue el programa de computador no comercial de rango más alto de los campeonatos mundiales de 1999.

Varias tentativas se han hecho para vencer los problemas «de la aproximación estándar» perfilados en la Sección 6.7. El primer algoritmo selectivo de búsqueda con un poco de base teórica fue probablemente B* (Berlinés, 1979), que intenta mantener límites de intervalos sobre el valor posible de un nodo en el árbol de juegos, más que darle una estimación valorada por un punto. Los nodos hoja son seleccionados para expansión en una tentativa de refinar los límites del nivel superior hasta que un movimiento sea «claramente mejor». Palay (1985) amplía la idea de B* para usar distribuciones de probabilidad en lugar de intervalos. La búsqueda del número de conspiración

sus propias cartas. Estas clases de comportamientos se generan automáticamente por un algoritmo óptimo para juegos de la información imperfecta. Tal algoritmo no busca en el espacio de estados del mundo (las manos de las cartas), sino en el espacio de **estados de creencia** (creencia de quién tiene qué cartas, con qué probabilidades). Seremos capaces de explicar el algoritmo correctamente en el Capítulo 17, una vez que hayamos desarrollado la maquinaria probabilística necesaria. En ese capítulo, ampliaremos también un punto final y muy importante: en juegos de información imperfecta, lo mejor es dar tan poca información al oponente como sea posible, y a menudo el mejor modo de hacerlo es actuar de manera *impredecible*. Por eso, los inspectores de sanidad hacen visitas de inspección aleatorias a los restaurantes.

6.6 Programas de juegos

Podríamos decir que jugar a juegos es a IA como el Gran Premio de carreras de automóviles es a la industria de coches: los programas de juegos son deslumbrantemente rápidos, máquinas increíblemente bien ajustadas que incorporan técnicas muy avanzadas de la ingeniería, pero no son de mucho uso para hacer la compra. Aunque algunos investigadores crean que jugar a juegos es algo irrelevante en la corriente principal de IA, se sigue generando entusiasmo y una corriente estable de innovaciones que se han adoptado por la comunidad.

AJEDREZ

Ajedrez: en 1957, Herbert Simon predijo que dentro de 10 años los computadores ganarían al campeón mundial humano. Cuarenta años más tarde, el programa Deep Blue derrotó a Garry Kasparov en un partido de exhibición a seis juegos. Simon se equivocó, pero sólo por un factor de 4. Kasparov escribió:

> El juego decisivo del partido fue el juego 2, que dejó una huella en mi memoria... Vimos algo que fue más allá de nuestras expectativas más salvajes de cómo un computador sería capaz de prever las consecuencias posicionales a largo plazo de sus decisiones. La máquina rechazó moverse a una posición que tenía una ventaja decisiva a corto plazo, mostrando un sentido muy humano del peligro. (Kasparov, 1997)

Deep Blue fue desarrollado por Murray Campbell, Feng-Hsiung Hsu, y Joseph Hoane en IBM (*véase* Campbell *et al*., 2002), construido sobre el diseño de Deep Thought desarrollado anteriomente por Campbell y Hsu en Carnegie Mellon. La máquina ganadora era un computador paralelo con 30 procesadores IBM RS/6000 que controlaba la «búsqueda *software*» y 480 procesadores VLSI, encargados para el ajedrez, que realizaron la generación de movimientos (incluso el movimiento de ordenación), la «búsqueda *hardware*» para los últimos niveles del árbol, y la evaluación de los nodos hoja. Deep Blue buscó 126 millones de nodos por segundo, como regla general, con una velocidad máxima de 330 millones de nodos por segundo. Generó hasta 30 billones de posiciones por movimiento, y alcanzó la profundidad 14 rutinariamente. El corazón de la máquina es una búsqueda alfa-beta estándar de profundidad iterativa con una tabla de transposiciones, pero la llave de su éxito parece haber estado en su capacidad de generar extensiones más allá del límite de profundidad para líneas suficientemente interesantes. En

algunos casos la búsqueda alcanzó una profundidad de 40 capas. La función de evaluación tenía más de 8.000 características, muchas de ellas describiendo modelos muy específicos de las piezas. Se usó una «salida de libro» de aproximadamente 4.000 posiciones, así como una base de datos de 700.000 jugadas de gran maestro de las cuales se podrían extraer algunas recomendaciones de consenso. El sistema también utilizó una gran base de datos de finales del juego de posiciones resueltas, conteniendo todas las posiciones con cinco piezas y muchas con seis piezas. Esta base de datos tiene el considerable efecto de ampliar la profundidad efectiva de búsqueda, permitiendo a Deep Blue jugar perfectamente en algunos casos aun cuando se aleja del jaque mate.

El éxito de Deep Blue reforzó la creencia de que el progreso en los juegos de computador ha venido principalmente del *hardware* cada vez más poderoso (animado por IBM). Los creadores de Deep Blue, por otra parte, declaran que las extensiones de búsqueda y la función de evaluación eran también críticas (Campbell *et al*., 2002). Además, sabemos que mejoras algorítmicas recientes han permitido a programas, que se ejecutan sobre computadores personales, ganar cada Campeonato Mundial de Ajedrez de computadores desde 1992, a menudo derrotando a adversarios masivamente paralelos que podrían buscar 1.000 veces más nodos. Una variedad de poda heurística se utiliza para reducir el factor de ramificación efectivo a menos de 3 (comparado con el factor de ramificación actual de aproximadamente 35). Lo más importante de ésta es la heurística de **movimiento nulo**, que genera una cota inferior buena sobre el valor de una posición, usando una búsqueda superficial en la cual el adversario consigue moverse dos veces. Esta cota inferior a menudo permite la poda alfa-beta sin el costo de una búsqueda de profundidad completa. También es importante la **poda de inutilidad**, la cual ayuda ha decidir de antemano qué movimientos causarán un corte beta en los nodos sucesores.

MOVIMIENTO NULO

PODA DE INUTILIDAD

El equipo de Deep Blue rehusó la posibilidad de un nuevo partido con Kasparov. En cambio, la competición principal más reciente de 2002 destacó el programa FRITZ contra el campeón mundial Vladimir Kramnik. El partido a ocho juegos terminó en un empate. Las condiciones del partido eran mucho más favorables al humano, y el *hardware* era un computador personal ordinario, no un supercomputador. De todos modos, Kramnik comentó que «está ahora claro que el programa y el campeón mundial son aproximadamente iguales».

DAMAS

Damas: en 1952, Arthur Samuel de IBM, trabajando en sus ratos libres, desarrolló un programa de damas que aprendió su propia función de evaluación jugando con él mismo miles de veces. Describimos esta idea más detalladamente en el Capítulo 21. El programa de Samuel comenzó como un principiante, pero después, sólo unos días de auto-jugar, se había mejorado más allá del propio nivel de Samuel (aunque él no fuera un jugador fuerte). En 1962 derrotó a Robert Nealy, un campeón en «damas ciegas», por un error por su parte. Muchas personas señalaron que, en damas, los computadores eran superiores a la gente, pero no era la cuestión. De todos modos, cuando uno considera que el equipo calculador de Samuel (un IBM 704) tenía 10.000 palabras de memoria principal, cinta magnetofónica para el almacenaje a largo plazo y un procesador de ,000001 GHz, el triunfo sigue siendo un gran logro.

Pocas personas intentaron hacerlo mejor hasta que Jonathan Schaeffer y colegas desarrollaran Chinook, que se ejecuta sobre computadores personales y usa la búsqueda alfa-

riamente pierde (−1). Esta oposición entre las funciones de utilidad de los agentes hace la situación entre adversarios. Consideraremos brevemente en este capítulo juegos multi-jugador, juegos de suma no cero, y juegos estocásticos, pero aplazaremos hasta el Capítulo 17 la discusión de la teoría de juegos apropiada.

Los juegos han ocupado las facultades intelectuales de la gente (a veces a un grado alarmante) mientras ha existido la civilización. Para los investigadores de IA, la naturaleza abstracta de los juegos los hacen un tema atractivo a estudiar. El estado de un juego es fácil de representar, y los agentes están restringidos, por lo general, a un pequeño número de acciones cuyos resultados están definidos por reglas precisas. Los juegos físicos, como croquet y hockey sobre hielo, tienen descripciones mucho más complicadas, una variedad mucho más grande de acciones posibles, y reglas bastante imprecisas que definen la legalidad de las acciones. A excepción del fútbol de robots, estos juegos físicos no han tenido mucho interés en la comunidad de IA.

El jugar a juegos fue una de las primeras tareas emprendidas en IA. Hacia 1950, casi tan pronto como los computadores se hicieron programables, el ajedrez fue abordado por Konrad Zuse (el inventor del primer computador programable y del primer lenguaje de programación), por Claude Shannon (el inventor de la teoría de información), por Norbert Wiener (el creador de la teoría de control moderna), y por Alan Turing. Desde entonces, hubo un progreso continuo en el nivel de juego, hasta tal punto que las máquinas han superado a la gente en las damas y en Otelo, han derrotado a campeones humanos (aunque no siempre) en ajedrez y *backgammon*, y son competitivos en muchos otros juegos. La excepción principal es Go, en el que los computadores funcionan a nivel aficionado.

Los juegos, a diferencia de la mayor parte de los problemas de juguete estudiados en el Capítulo 3, son interesantes *porque* son demasiado difíciles para resolverlos. Por ejemplo, el ajedrez tiene un factor de ramificación promedio de aproximadamente 35, y los juegos a menudo van a 50 movimientos por cada jugador, entonces el árbol de búsqueda tiene aproximadamente 35^{100} o 10^{154} nodos (aunque el grafo de búsqueda tenga «sólo» aproximadamente 10^{40} nodos distintos). Por lo tanto, los juegos, como el mundo real, requieren la capacidad de tomar *alguna* decisión cuando es infactible calcular la decisión *óptima*. Los juegos también castigan la ineficiencia con severidad. Mientras que una implementación de la búsqueda A* que sea medio eficiente costará simplemente dos veces más para ejecutarse por completo, un programa de ajedrez que sea medio eficiente en la utilización de su tiempo disponible, probablemente tendrá que descartarse si no intervienen otros factores. La investigación en juegos ha generado, por lo tanto, varias ideas interesantes sobre cómo hacer uso, lo mejor posible, del tiempo.

Comenzamos con una definición del movimiento óptimo y un algoritmo para encontrarlo. Veremos técnicas para elegir un movimiento bueno cuando el tiempo es limitado. La **poda** nos permite ignorar partes del árbol de búsqueda que no marcan ninguna diferencia para obtener la opción final, y las **funciones de evaluación** heurísticas nos permiten aproximar la utilidad verdadera de un estado sin hacer una búsqueda completa. La Sección 6.5 habla de juegos como el backgammon que incluyen un elemento de posibilidad; también hablamos del bridge, que incluye elementos de **información imperfecta** al no ser visibles todas las cartas a cada jugador. Finalmente, veremos cómo se desenvuelven los programas de juegos contra la oposición humana y las direcciones para el desarrollo futuro.

INFORMACIÓN IMPERFECTA

ramificación es 1. (Se puede pensar que la búsqueda de estabilidad es una variante de extensiones excepcionales.) En la Figura 6.9, una búsqueda de extensión excepcional encontrará el movimiento de convertirse en reina, a condición de que los movimientos de jaque de Negro y los movimientos del rey blanco puedan identificarse como «claramente mejores» que las alternativas.

Hasta ahora hemos hablado del corte de la búsqueda a un cierto nivel y que la poda alfa-beta, probablemente, no tiene ningún efecto sobre el resultado. Es también posible hacer la **poda hacia delante**, en la que podamos inmediatamente algunos movimientos de un nodo. Claramente, la mayoría de la gente que juega al ajedrez sólo considera unos pocos movimientos de cada posición (al menos conscientemente). Lamentablemente, esta aproximación es bastante peligrosa porque no hay ninguna garantía de que el mejor movimiento no sea podado. Esto puede ser desastroso aplicado cerca de la raíz, porque entonces a menudo el programa omitirá algunos movimientos «evidentes». La poda hacia delante puede usarse en situaciones especiales (por ejemplo, cuando dos movimientos son simétricos o equivalentes, sólo consideramos uno) o para nodos profundos en el árbol de búsqueda.

PODA HACIA DELANTE

La combinación de todas las técnicas descritas proporciona un programa que puede jugar al ajedrez loablemente (o a otros juegos). Asumamos que hemos implementado una función de evaluación para el ajedrez, un test del límite razonable con una búsqueda de estabilidad, y una tabla de transposiciones grande. También asumamos que, después de meses de intentos tediosos, podemos generar y evaluar alrededor de un millón de nodos por segundo sobre los últimos PCs, permitiéndonos buscar aproximadamente 200 millones de nodos por movimiento bajo un control estándar del tiempo (tres minutos por movimiento). El factor de ramificación para el ajedrez es aproximadamente 35, como media, y 35^5 son aproximadamente 50 millones, así si usamos la búsqueda minimax podríamos mirar sólo cinco capas. Aunque no sea incompetente, tal programa puede ser engañado fácilmente por un jugador medio de ajedrez, el cual puede planear, de vez en cuando, seis u ocho capas. Con la búsqueda alfa-beta nos ponemos en aproximadamente 10 capas, y resulta un nivel experto del juego. La Sección 6.7 describe técnicas de poda adicionales que pueden ampliar la profundidad eficaz de búsqueda a aproximadamente 14 capas. Para alcanzar el nivel de gran maestro necesitaríamos una función de evaluación ajustada y una base de datos grande de movimientos de apertura y movimientos finales óptimos. No sería malo tener un supercomputador para controlar este programa.

6.5 Juegos que incluyen un elemento de posibilidad

En la vida real, hay muchos acontecimientos imprevisibles externos que nos ponen en situaciones inesperadas. Muchos juegos reflejan esta imprevisibilidad con la inclusión de un elemento aleatorio, como el lanzamiento de dados. De esta manera, ellos nos dan un paso más cercano a la realidad, y vale la pena ver cómo afecta al proceso de toma de decisiones.

Backgammon es un juego típico que combina la suerte y la habilidad. Se hacen rodar unos dados, al comienzo del turno de un jugador, para determinar los movimientos

de poda selectivos para vencer el enorme factor de ramificación. Zobrist (1970) usó las reglas condición-acción para sugerir movimientos plausibles cuando aparecieran los modelos conocidos. Reitman y Wilcox (1979) combinan reglas y búsqueda con efectos buenos, y los programas más modernos han seguido esta aproximación híbrida. Müller (2002) resume el estado del arte de la informatización de Go y proporciona una riqueza de referencias. Anshelevich (2000) utilizó las técnicas relacionadas para el juego Hex. *Computer Go Newsletter*, publicada por la Asociación de Go por computador, describe el desarrollo actual del juego.

Los trabajos sobre juegos de computador aparecen en multitud de sitios. La mal llamada conferencia *Heuristic Programming in Artificial Intelligence* hizo un informe sobre las Olimpiadas de Computador, que incluyen una amplia variedad de juegos. Hay también varias colecciones de trabajos importantes sobre la investigación en juegos (Levy, 1988a, 1988b; Marsland y Schaeffer, 1990). La Asociación Internacional de Ajedrez por Computador (ICCA), fundada en 1977, publica la *revista* trimestral *ICGA* (anteriormente la *revista ICCA*). Los trabajos importantes han sido publicados en la serie antológica *Advances in Computer Chess*, que comienza con Clarke (1977). El volumen 134 de la revista *Artificial Intelligence* (2002) contiene descripciones de programas para el ajedrez, Otelo, Hex, shogi, Go, *backgammon*, poker, Scrabble™ y otros juegos.

EJERCICIOS

6.1 En este problema se ejercitan los conceptos básicos de juegos, utilizando tic-tac-toe (tres en raya) como un ejemplo. Definimos X_n como el número de filas, columnas, o diagonales con exactamente n *Xs* y ningún O. Del mismo modo, O_n es el número de filas, columnas, o diagonales con solamente n *Os*. La función de utilidad asigna $+1$ a cualquier posición con $X_3 = 1$ y -1 a cualquier posición con $O_3 = 1$. Todas las otras posiciones terminales tienen utilidad 0. Para posiciones no terminales, usamos una función de evaluación lineal definida como $Eval(s) = 3\,X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$.

- ***a***) ¿Aproximadamente cuántos juegos posibles de tic-tac-toe hay?
- ***b***) Muestre el árbol de juegos entero hasta profundidad 2 (es decir, un *X* y un *O* sobre el tablero) comenzando con un tablero vacío, teniendo en cuenta las simetrías.
- ***c***) Señale sobre el árbol las evaluaciones de todas las posiciones a profundidad 2.
- ***d***) Usando el algoritmo minimax, marque sobre su árbol los valores hacia atrás para las posiciones de profundidades 1 y 0, y use esos valores para elegir el mejor movimiento de salida.
- ***e***) Marque los nodos a profundidad 2 que no serían evaluados si se aplicara la poda alfa-beta, asumiendo que los nodos están generados *en orden óptimo por la poda alfa-beta*.

6.2 Demuestre la afirmación siguiente: para cada árbol de juegos, la utilidad obtenida por MAX usando las decisiones minimax contra MIN subóptimo nunca será inferior que la utilidad obtenida jugando contra MIN óptimo. ¿Puede proponer un árbol de juegos en el cual MAX puede mejorar utilizando una estrategia subóptima contra MIN subóptimo?