Ciclos hamiltonianos del caballo de ajedrez en un tablero 8×8

por

Alberto Bañón Serrano

Resumen. Un mecanismo de poda original y muy eficiente incorporado a una aplicación informática de búsqueda por el método de «vuelta atrás» (*backtracking*) ha permitido recorrer —creemos que por primera vez— el árbol correspondiente a los recorridos del caballo en un tablero de ajedrez estándar (8×8) y encontrar todos los recorridos cerrados (ciclos hamiltonianos). El número de ciclos obtenidos, 13 267 364 410 532, coincide con el calculado, mediante un ingenioso método, por Brendan Mackay en el año 1997, sin que tengamos constancia de que haya sido confirmado computacionalmente hasta ahora. El mecanismo de poda se puede aplicar, en general, a otros problemas de ciclos hamiltonianos.

Introducción

El problema del salto del caballo en un tablero de ajedrez, que consiste en recorrer las 64 casillas pasando una sola vez por cada una de ellas, está resuelto desde principios del siglo IX, en el sentido de que se conocen algunas soluciones (figura 1).

Pero han sido muchos los matemáticos que han dedicado tiempo a tratar de entenderlo, entre ellos, Leonhard Euler (1707–1783). Euler publicó un extenso trabajo en el año 1759 en el que describe, fundamentalmente, dos estrategias. La primera muestra cómo transformar un recorrido incompleto en uno completo, es decir, uno que no recorre las 64 casillas en otro que sí lo hace. La otra sirve para transformar un recorrido abierto en cerrado. Un recorrido cerrado es aquel que termina en una casilla desde la que el caballo puede saltar a la casilla con la que se inició el recorrido. Ed Sandifer describe los detalles en [4].

En la historia de este problema, hay que destacar un método para encontrar soluciones desarrollado por H. C. Warnsdorf [5] en el siglo XIX. La regla consiste en ir moviendo el caballo a la casilla desde la que puede realizar el mínimo número de saltos. El objetivo es evitar que el caballo se atasque porque llega a una casilla desde la que no puede mover a ninguna que no haya visitado antes. Esta regla se entiende mejor si pensamos a la inversa, si en lugar de a cuántas casillas podemos ir desde una dada, lo vemos como desde cuántas casillas se puede llegar a ella. Hay que visitar primero las casillas que con más probabilidad se quedarían aisladas cuando el caballo empiece a realizar su recorrido; una casilla a la que sólo se puede llegar desde dos, se quedará aislada el doble de pronto que una a la que se puede llegar desde cuatro.

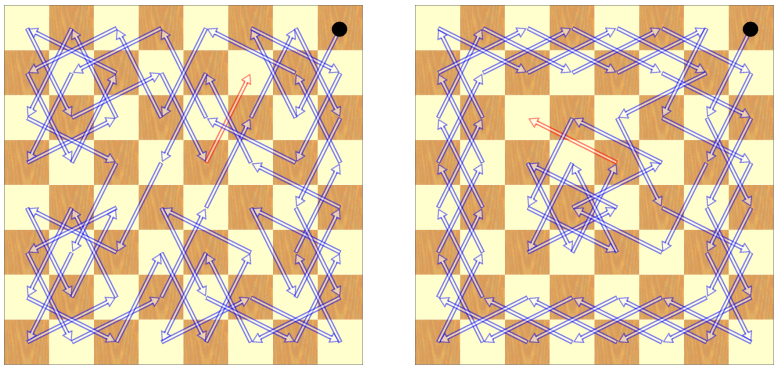

Figura 1: A la izquierda, la primera solución publicada al problema del salto del caballo. Aparece en el *Kitab ash-Shatranj* (Libro del ajedrez), escrito por al-Adli ar-Rumi (ca. 840). Ha llegado a nosotros a través del manuscrito de Abu Zakariya Yahya ben Ibrahim al-Hakim (ca. 1350) titulado *Nuzhat al-arbab al-'aqulfi'sh-shatranj al-manqul* (El deleite del inteligente, una descripción del ajedrez) [3]. En el mismo manuscrito figura la solución de la derecha, sin datar (podría ser anterior a la de al-Adli) y atribuida a un por lo demás desconocido Ali C. Mani. Obsérvese que la solución de ar-Rumi puede cerrarse volviendo a la casilla inicial.

Esta idea es básica en este trabajo, no para buscar soluciones, sino para todo lo contrario: detectar, cuanto antes, cuándo no se encontrará una.

En 1995, Martin Löbbing e Ingo Wegener [1] se sirvieron de 20 estaciones de trabajo durante cuatro meses y concluyeron que había alrededor de treinta y tres billones de soluciones (exactamente 33 439 123 484 294) al problema del salto del caballo. Dos años después se encontró un error en la implementación del modelo matemático y Brendan McKay [2], por un procedimiento distinto, redujo la cifra a aproximadamente 13 billones (13 267 364 410 532).

1. METODOLOGÍA

Aunque este trabajo se ha centrado exclusivamente en el caso del tablero de ajedrez estándar, de tamaño 8×8, el problema del salto del caballo se ha estudiado para tableros con dimensiones arbitrarias. Por ejemplo, se ha demostrado que para un tablero de ajedrez $m \times n$, con $m \leq n$, el problema del caballo de ajedrez no tiene solución (o, con lenguaje que introduciremos en breve, no existen ciclos hamiltonianos para el grafo del salto del caballo) si se cumple alguna de las tres condiciones:

1. m y n son ambos impares;
2. $m = 1$, 2 o 4;
3. $m = 3$ y $n = 4$, 6 u 8.

El origen de la teoría de grafos se remonta al siglo XVIII con el problema de los puentes de Königsberg, el cual consistía en encontrar un camino que recorriera los siete puentes del río en la ciudad de Königsberg, de modo que se recorrieran todos

los puentes pasando una sola vez por cada uno de ellos. El trabajo que Euler publicó en 1736 sobre este problema es considerado el primer resultado de la teoría de grafos.

El problema del salto de caballo es distinto al de los puentes de Königsberg ya que son los vértices, en lugar de las aristas, los que se exige que se recorran una sola vez. Este tipo de caminos se denominan *hamiltonianos* en honor a W. R. Hamilton, que en el año 1857 resolvió el problema de recorrer todos los vértices de un dodecaedro sin pasar dos veces por ninguno de ellos.

Los vértices del grafo correspondiente al salto del caballo son las 64 casillas del tablero, y las aristas cualquier línea que unas dos casillas válidas para ser la casilla inicial y final de un movimiento legal del caballo. Un caballo puede mover a cualquier casilla de color distinto a la que ocupa y que diste dos casillas de la de partida (se suele decir que *mueve en L*). Este cambio en el color de la casilla al mover el caballo convierte al grafo del salto del caballo en un grafo bipartito.

Decidir si un grafo posee un camino hamiltoniano es un problema NP-completo, es decir, difícil de resolver pero fácil de comprobar, lo que implica que casi seguro no existe ningún algoritmo general para resolver el problema en tiempo polinómico. La solución requiere básicamente evaluar todas las posibilidades, dando lugar a un orden de complejidad exponencial o factorial.

Los algoritmos generales de complejidad exponencial para un grafo con 64 vértices, como el del salto del caballo, requieren tiempos totalmente fuera del alcance de la capacidad actual de cálculo, por lo que en este trabajo se ha encontrado una solución ad hoc. En particular, probar todos los recorridos posibles es inabordable con nuestros ordenadores, ya que su número es del orden de 10^{24}, de acuerdo con la siguiente estimación.

Para el primer salto, dependiendo de la casilla de la que se parta, se dispone de entre 2 y 8 posibilidades. Considerando las 10 casillas iniciales que son independientes (no hay simetrías entre ellas), para el primer salto hay 52 posibilidades y para el salto 16 el número asciende a 158 228 932 546. Este número sigue creciendo con el número de saltos, pero según se va llegando al salto 63 su tasa de crecimiento debe aproximarse a la unidad porque cada vez quedan menos casillas libres. La figura 2 muestra la tasa de crecimiento calculada de forma exacta para los 16 primeros saltos y establecida igual a 1 para los saltos 62 y 63, que es lo que realmente ocurre ya que en los saltos 62 y 63 hay el mismo número de casillas disponibles, en concreto una, y por tanto el factor respecto al salto anterior es 1.

Ajustando los datos de la figura 2 a un polinomio de grado dos, mediante mínimos cuadrados, podemos interpolar la tasa de crecimiento para los saltos 17 a 61 y con ello estimar el número de recorridos con 64 saltos, obteniéndose la cifra, ya mencionada, de aproximadamente 10^{24} caminos hamiltonianos.

Con esta cifra hay que descartar la idea de intentar todos los recorridos posibles y ver cuáles acaban recorriendo las 64 casillas. Pero no es necesario probarlos todos: podemos buscar reglas que nos digan, con certeza absoluta, cuándo un recorrido terminará inevitablemente atascado. La bondad de la regla vendrá determinada por el grado de anticipación con el que nos indique que el camino que estamos recorriendo no llegará a una solución. Obviamente, cuanto antes lo sepamos menos tiempo emplearemos en continuar inútilmente.

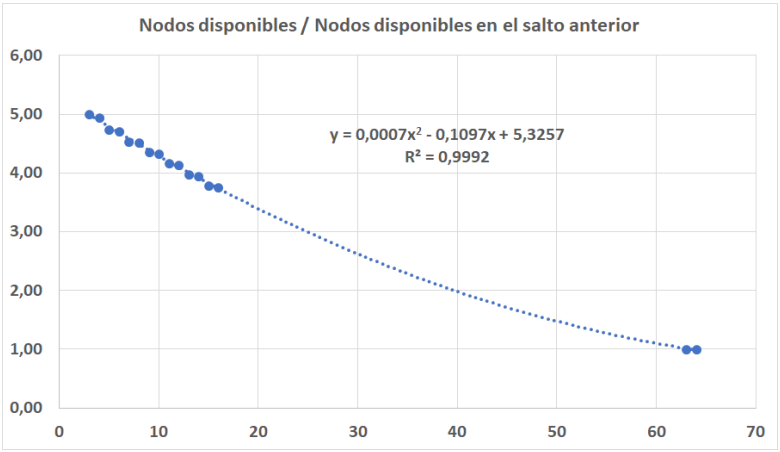

Figura 2: Tasa de crecimiento del número de nodos disponibles para cada salto.

La idea de Warnsdorf es mantener una tabla 8×8 con el número de casillas a las que se puede saltar desde cada casilla. Es evidente que el número de casillas a las que se puede saltar, desde una dada, es el mismo que el número de casillas desde las que se puede llegar hasta ella. Este cambio de sentido en la interpretación de los números de la tabla es lo que la hace útil en este trabajo, porque lo que interesa es saber cuándo aparecen casillas a las que no se puede llegar. En lo sucesivo, a la tabla la llamaremos *tabla W*, y el número que aparece en cada casilla será su *valor W*. Para referirnos a ellas, numeraremos las casillas de abajo a arriba y de izquierda a derecha.

2	3	4	4	4	4	3	2
3	4	6	6	6	6	4	3
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	**6**	8	8	8	8	6	4
3	4	**6**	6	6	6	4	3
2	3	4	4	4	4	3	2

Figura 3: Tabla Warnsdorf antes de posar el caballo sobre el tablero.

El 2 en la casilla del vértice inferior izquierdo (la casilla 1) de la figura 3 significa que desde ella se puede saltar a dos casillas, o que se puede llegar a ella desde dos casillas distintas, las número 11 y 18, marcadas en negrita en dicha tabla.

Una vez que se ejecuta el salto desde la casilla uno a la once (figura 4), la casilla 18 ya no se podrá alcanzar desde la 1 y su valor W pasa de 6 a 5.

2	3	4	4	4	4	3	2
3	4	6	6	6	6	4	3
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	**5**	8	8	8	8	6	4
3	4	**H**	6	6	6	4	3
D	3	4	4	4	4	3	2

Figura 4: Tabla W después del salto de la casilla 1 (D) a la 11 (H).

1.1. Casillas en isla

Pues bien, en el momento en que en una casilla aparezca un cero, no hay que seguir con el recorrido del caballo: aunque pueda seguir saltando, la casilla del cero nunca podrá ser alcanzada, es una isla. Mostramos un ejemplo en la figura 5.

x	x	x	4	x	x	x	2
x	4	x	x	x	5	x	x
x	x	4	5	5	x	4	x
4	x	***5***	x	6	4	5	x
X	x	4	***5***	3	5	x	4
X	**H**	x	4	3	6	x	3
0	x	x	x	x	x	4	x
E	x	**D**	x	3	x	2	x

Figura 5: La casilla 9 es una isla.

Después de 37 saltos desde la casilla 1 (E) —las casillas ya visitadas están marcadas con una x—, donde el último salto fue de la casilla 3 (D) a la 18 (H), la casilla 9 (0) se ha vuelto inaccesible, ya que en el paso anterior sólo se podía llegar a ella desde la 3. Desde la casilla 18 (H) aún se puede seguir saltando a cualquiera de las casillas 28, 33 o 35 (señaladas en itálica) y desde estas a otras, pero no llegaríamos a ninguna solución a nuestro problema. Incorporar esta regla a la aplicación informática reduce el tiempo de cálculo a la mitad, importante pero insuficiente.

La razón de este escaso beneficio es que el futuro no queda predeterminado hasta muy cerca del final (*el atasco*). En la mayoría de los casos basta con cambiar el último movimiento realizado, o el anterior a este, para que ese recorrido ligeramente modificado sí pueda llegar a una solución. Nunca encontraremos una regla que nos permita detenernos mientras el atasco sea evitable.

1.2. Casillas colgantes

La idea que hace realmente posible recorrer el árbol en un tiempo asequible es usar las casillas con valor W igual a 1, que llamaremos *colgantes*, ya que únicamente se puede acceder a ellas desde una casilla. Esto significa que, de existir una solución, esa sería la última casilla del recorrido: podemos llegar a ella, pero después no habrá a donde saltar.

Si añadimos que el caballo cambia de color a cada salto —si parte de una casilla blanca, va a una negra, y viceversa—, podemos adelantar que, si el recorrido se inicia en una casilla blanca, debe terminar en una negra (63 saltos es un número impar). Por tanto, si la casilla colgante es del mismo color que la casilla de partida podemos dejar de saltar, ya que sabemos que no completaremos el recorrido. Esta regla consigue dividir por cinco el tiempo de cálculo.

x	x	x	4	x	x	x	**1**
x	4	x	x	x	**5**	x	x
x	x	4	5	4	x	**D**	x
4	x	5	x	5	4	5	x
x	x	4	6	4	x	x	**H**
x	4	3	4	3	6	x	2
2	x	x	x	4	x	3	x
E	x	4	x	3	x	3	x

Figura 6: La casilla 64 es una casilla colgante.

En la figura 6 vemos un ejemplo. Empezando en la casilla 1 (E), tras el salto número 36, de la casilla 47 (D) a la 32 (H), la casilla 64 (con valor W igual a 1, marcado en negrita) queda colgando de la 54 (donde aparece un 5 en negrita). La casilla 64 es del mismo color que la 1, por lo que no puede ser la última de un recorrido con un número impar de saltos.

Además, la casilla colgante ofrece un beneficio añadido. Si el caballo consigue dar el salto 61 sin que haya actuado el filtro de la casilla colgante, podemos asegurar que completará el recorrido hasta la 64. El valor W de la casilla ocupada en el salto 61 será, necesariamente, 2 o 3. Si es 2, hay una solución recorriendo una tras otra, las tres casillas que aún están vacías y, si el valor W es 3, hay dos soluciones porque

las tres casillas vacías, junto a la ocupada en el salto 61, forman un rombo que se puede recorrer en el sentido de las agujas del reloj y en el contrario.

1.3. Reglas ineficaces

Parece tentador intentar exprimir aún más las casillas colgantes. La idea es anotar cuándo se crea una casilla colgante (su valor W se hace igual a 1) de las que no detienen la secuencia de saltos, porque es de distinto color que la casilla de partida. Luego, cada vez que demos un salto, miramos si la casilla anotada está entre los posibles destinos. De ser así, la secuencia ha terminado, pues la casilla colgante se convertirá en isla o, si la ocupamos en este salto, no podremos seguir saltando. Pero es más el coste de hacer este seguimiento que el que se evita. Este filtro es casi equivalente al control de las casillas en isla. De hecho, si se implementa nunca se llegaría a tener una casilla en isla, y el coste de comprobar las casillas en isla es menor que el de las colgantes. También se podría intentar controlar la aparición de una segunda casilla colgante que hiciese imposible llegar a una solución, pero es una situación muy improbable que no justifica modificar el algoritmo para lograr un menor coste.

Perseverando en la idea de predecir cuándo un recorrido no podrá alcanzar las 64 casillas, podemos asegurar que esto es lo que ocurrirá si lo ya recorrido «divide el tablero» en dos partes aisladas con, al menos, una casilla vacía en cada orilla, como ilustra la figura 7. Si ya se han recorrido las casillas marcadas con una x, además de algunas otras para que esto sea posible, el caballo no puede pasar de la parte izquierda a la derecha, o viceversa, por lo que en el momento que esto ocurra podemos abandonar el recorrido.

			x	x			
			x	x			
			x	x			
			x	x			
			x	x			
			x	x			
			x	x			
			x	x			

Figura 7: Tablero dividido.

Pero resulta que, una vez más, comprobar la formación de esa estructura, o de sus simétricas, es más costoso, en tiempo de cálculo, que el beneficio que se obtiene.

El motivo de que estas estrategias no aporten valor es que los signos de alarma no aparecen hasta que el atasco está muy cerca, tanto que es preferible seguir hasta consumarlo. Aunque el coste informático de la comprobación sea muy pequeño, como hay que hacerla en cada uno de los pasos de cada uno de los intentos de recorrido, se convierte en un coste más importante que el propio avance por el árbol hasta el atasco. Además, unas estrategias solapan a otras. Por ejemplo, muchos de los

recorridos interrumpidos por el filtro del «tablero dividido» se habrían interrumpido igualmente un poco después por el de las «casillas colgantes». Esto hace que la utilidad del primero sea poca, existiendo el segundo con menos coste de cálculo.

1.4. SUBRECORRIDOS EQUIVALENTES

Diremos que dos «subrecorridos» con el mismo número de saltos son *equivalentes* si terminan en la misma casilla y si previamente también han recorrido las mismas casillas, quizás en órdenes distintos. Está claro que el número de soluciones que empiezan con dos subrecorridos equivalentes es el mismo. De hecho, las soluciones son las mismas a partir de la casilla común en la que terminan ambos subrecorridos.

Si tabulamos el número de soluciones que se obtienen empezando con todos los subrecorridos de un determinado número de saltos, aprovechar los subrecorridos equivalentes nos permitirá evitar muchas búsquedas. Podemos pedir a la aplicación que controle subrecorridos de, por ejemplo, 20 saltos, sin más que pedirle que al llegar al salto 20 cree una «clave» que sea un número binario de 64 bits con un 1 en las posiciones correspondientes a las casillas ocupadas y un 0 en las restantes. Todos los subrecorridos equivalentes están representados por la misma clave, y un ordenador (en particular si la aplicación es de 64 bits) maneja muy rápidamente claves como estas. Para que buscar una clave sea lo más eficiente posible (será una búsqueda dicotómica), nuestra aplicación crea un árbol binario con las claves según van apareciendo —o más exactamente, para tener en cuenta la casilla a la que se llegó en el último salto, 64 árboles, uno por casilla—.

Cuando la aplicación completa, siguiendo con el ejemplo, el salto 20, crea la clave con las casillas ocupadas hasta aquí y busca en el árbol binario que corresponda a la casilla de llegada si esa clave ya existe. Si es así, no tiene que seguir profundizando en todos los recorridos posibles a partir de ese punto (que son muchísimos), porque la aplicación habrá guardado cuántas soluciones se pueden encontrar a partir de aquí la primera vez que exploró un subrecorrido equivalente al actual.

A mayor número de saltos, mayor ganancia, pero el tamaño de la tabla —que está limitado por la RAM del ordenador— crece exponencialmente con el número de saltos: para tabular todos los nodos a una profundidad de 20 saltos se necesitan del orden de 5 Gbytes. Por ello es más eficiente trabajar a mayor profundidad, pero reiniciar la tabla desde cero cuando se llena. Esto, que puede resultar sorprendente, funciona porque, por un lado, los recorridos que nos hayamos ahorrado explorar antes de reiniciar la tabla, ahorrados quedan. Por otro, como los posibles recorridos se generan por *backtraking*, es más probable encontrar subrecorridos equivalentes en un rango de intentos sucesivos que no en intentos realizados mucho antes. Por eso no es tan grave eliminar de los árboles la información «antigua». Con esta estrategia, las pruebas realizadas indican que se alcanza la mayor eficiencia para subrecorridos en el entorno de 30 saltos.

1.5. El orden de los saltos

Otro factor que se ha analizado es el orden en que se examinan los nodos de cada uno de los niveles del árbol de recorridos. Al recorrer los nodos según la regla de Warnsdorf —recordemos que se trata de ir moviendo el caballo a la casilla desde la que puede realizar el mínimo número de saltos—, la velocidad a la que se encuentran las primeras soluciones es mucho mayor que si estos se recorren siguiendo un orden arbitrario. Pero no está claro qué ocurriría en una exploración exhaustiva del árbol. Aunque se encuentren todas las soluciones muy al principio, la búsqueda debe continuar hasta completar el árbol.

La aplicación dispone de una tabla que, para cada casilla, contiene todos los saltos que un caballo puede realizar desde ella. Inicialmente, el orden de los saltos en esta tabla estaba prefijado de forma arbitraría, en concreto, siguiendo el movimiento de las agujas del reloj.

El siguiente paso fue aplicar la regla de Warnsdorf y volver a ordenar los posibles saltos cada vez que se desciende en el árbol. Pero esto hay que hacerlo billones de veces y el gasto de tiempo es tan grande que, en lugar de reducirse el tiempo de búsqueda, se duplica. Una aproximación de compromiso es utilizar una tabla de saltos ordenada inicialmente por aplicación de la regla de Warnsdorf al tablero vacío y conservar está ordenación durante todo el cálculo. El resultado ha sido que el tiempo se reduce en torno al 10 %.

1.6. Soluciones cerradas

A pesar de todas las mejoras, los resultados obtenidos (en un PC con Windows 10 de 64 bits y una CPU Intel i7-3930K a 3,20 GHz) mostraban que se tardaría años en encontrar todas las soluciones, por lo que nos hemos centrado sólo en las soluciones cerradas (*ciclos*).

La búsqueda de soluciones cerradas es posible gracias a dos circunstancias:

1. El número de soluciones cerradas es el mismo, sea cual sea la casilla del tablero desde la que empecemos a saltar.
2. Las posibles casillas finales del recorrido, antes de volver a la inicial, son muy pocas, ya que desde ellas se tiene que poder saltar a la casilla en la que empezamos.

Aunque el método se ha presentado de forma específica para el problema del salto del caballo, puede ser útil para la enumeración de ciclos hamiltonianos en otros grafos: la idea central de podar el árbol mediante la tabla W tiene carácter general.

2. Cálculos

Las distintas ramas del árbol de recorridos que surge al considerar los posibles saltos que se pueden realizar desde cada una de las casillas del tablero son independientes entre sí, por lo que se pueden explorar de forma paralela si se dispone de

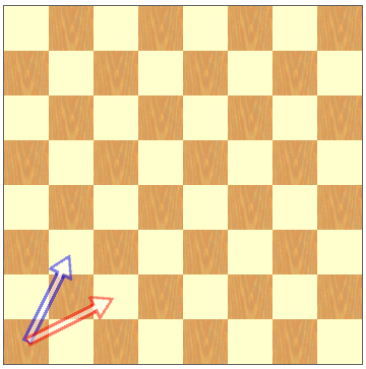

Figura 8: Inicio de los saltos desde la casilla 1.

varios ordenadores o de un ordenador con varios procesadores. A cada una de estas ramas las denominaremos *fracciones*.

Una casilla particularmente sencilla para iniciar los saltos es cualquiera de las cuatro esquinas, por ejemplo, la del vértice inferior izquierdo. Desde esta casilla sólo se puede saltar a dos, que hemos numerado como las casillas 11 y 18 (figura 8).

Esto significa que los recorridos cerrados empezarán por una de esas dos casillas y terminarán en la otra. Es evidente que el número de soluciones es el mismo empecemos por una o por otra. Es más, desde el punto de vista del grafo, en el que no importa el sentido en que se recorre el ciclo, se tienen de hecho las mismas soluciones: por cada solución cerrada que comienza con el salto a la casilla 18 para terminar en la 11, hay otra solución, si la recorremos en sentido contrario, empezando con el salto a la casilla 11 y terminando en la 18. Aquí se ha elegido empezar los recorridos con el salto a la casilla 18 para acabar con un salto a la 11.

Que la casilla final del recorrido esté predeterminada —partiendo de la 18 ha de ser necesariamente la 11— acelera mucho la velocidad de búsqueda al combinarla con la «regla de la casilla colgante». En el caso general, sabemos que cuando se produzca una casilla colgante del mismo color que la de partida, no habrá solución. Esto sigue siendo válido, pero ahora podemos ser más exigentes. Si desde la casilla colgante no se puede saltar a la 11, no habrá solución; la velocidad de exploración se multiplica por 10.

Partiendo de la casilla 1, con el primer salto a la casilla 18 y fijando de antemano los dos saltos siguientes, surgen 24 fracciones (cuadro 1), un número bastante adecuado para el ordenador utilizado en este trabajo, equipado con una CPU Intel Xeon ES-2630 v4 a 2,20 GHz (20 procesadores físicos, 40 lógicos).

Cuadro 1: Fracciones desde la casilla 1.

Inicial	**Salto 1**	**Salto 2**	**Salto 3**	**Último salto**
1	18	3	9	11
1	18	3	13	11

Cuadro 1: (continúa)

Inicial	**Salto 1**	**Salto 2**	**Salto 3**	**Último salto**
1	18	3	20	11
1	18	12	2	11
1	18	12	6	11
1	18	12	22	11
1	18	12	27	11
1	18	12	29	11
1	18	28	13	11
1	18	28	22	11
1	18	28	34	11
1	18	28	38	11
1	18	28	43	11
1	18	28	45	11
1	18	33	27	11
1	18	33	43	11
1	18	33	50	11
1	18	35	20	11
1	18	35	25	11
1	18	35	29	11
1	18	35	41	11
1	18	35	45	11
1	18	35	50	11
1	18	35	52	11

Al tener prefijados tres saltos y la casilla final, el árbol a recorrer es de profundidad 60 y no 64. Por el contrario, desde cualquier casilla que no sea una esquina, no se tiene una casilla final única y no se pueden prefijar tres saltos sin obtener un número de fracciones muy superior a 50. Podría parecer que la casilla 1 (o cualquiera de las otras 3 esquinas) es la que menos tiempo de cálculo necesitaría, pero se está olvidando un factor esencial: el reparto de soluciones por fracción. Lo óptimo sería que todas las fracciones tuviesen el mismo número de soluciones, de modo que las búsquedas en todas ellas terminasen a la vez y durante todo el tiempo de cálculo tuviésemos permanentemente ocupados a todos los procesadores del ordenador. Si una sola fracción acapara un número de soluciones muy superior a las restantes, se terminaría la exploración de todas menos esa, que continuaría su cálculo haciendo uso de un solo procesador, quedando ociosos los restantes. Si el número de soluciones de esa primera fracción es el doble que el de las otras, el tiempo de cálculo sería el doble que el necesario si todas las fracciones conducen al mismo número de soluciones. A priori, no se podía saber cómo de homogéneas eran las 24 fracciones seleccionadas a partir de la casilla 1, pero el avance de los cálculos reveló una dispersión enorme; tanto, que el tiempo que se estimó inicialmente en 50 días, apuntaba a una cifra cuatro veces mayor. El contrapunto lo puso la exploración a partir de la casilla 28.

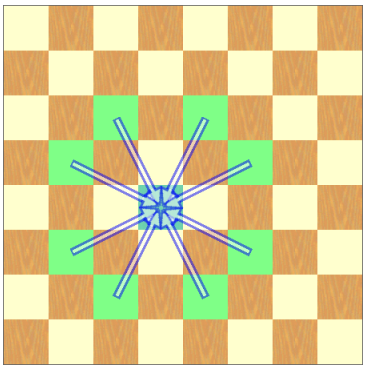

Figura 9: Casillas finales partiendo de la casilla 28.

Aunque sólo se necesitaba explorar los saltos a partir de una casilla (se eligió la 1), se programó hacerlo también a partir de otra (se eligió la 28) para comprobar que se llegaba a los mismos resultados y reforzar la confianza en la aplicación informática. Seleccionar las fracciones a partir de la casilla 28 es algo más laborioso que a partir de la 1, pero vale la pena. Las soluciones cerradas pueden terminar en cualquiera de las ocho casillas de la figura 9, y no sólo en dos, como es el caso cuando se parte de la casilla 1.

Sin prefijar ningún salto ya tenemos 8 fracciones atendiendo a la casilla final —en realidad 7, porque el primer salto es a una de esas mismas casillas y, por lo tanto, no puede ser a la vez el primero y el último—. Si prefijamos el primer salto, surgen $8 \times 7 = 56$ fracciones: 8 posibilidades para el salto inicial por 7 posibilidades para la casilla final (cuadro 2).

Cuadro 2: Fracciones desde la casilla 28.

Inicial	**Salto 1**	**Último salto**	**Inicial**	**Salto 1**	**Último salto**
28	11	13	28	34	11
28	11	18	28	34	13
28	11	22	28	34	18
28	11	34	28	34	22
28	11	38	28	34	38
28	11	43	28	34	43
28	11	45	28	34	45
28	13	11	28	38	11
28	13	18	28	38	13
28	13	22	28	38	18
28	13	34	28	38	22
28	13	38	28	38	34
28	13	43	28	38	43
28	13	45	28	38	45
28	18	11	28	43	11

Cuadro 2: (continúa)

Inicial	Salto 1	Último salto	Inicial	Salto 1	Último salto
28	18	13	28	43	13
28	18	22	28	43	18
28	18	34	28	43	22
28	18	38	28	43	34
28	18	43	28	43	38
28	18	45	28	43	45
28	22	11	28	45	11
28	22	13	28	45	13
28	22	18	28	45	18
28	22	34	28	45	22
28	22	38	28	45	34
28	22	43	28	45	38
28	22	45	28	45	43

Observemos que la mitad de estas 56 fracciones es redundante, ya que conduce a las mismas soluciones que las otras 28, pero recorridas en sentido contrario. Además, la fracción 28-11-...-18, y su opuesta 28-18-...-11 no contienen ninguna solución porque la casilla 1 queda en isla, por lo que prescindiremos de ellas a partir de ahora.

Así pues, podemos considerar, al iniciar los saltos desde la casilla 28, únicamente las 27 fracciones que muestra el cuadro 3 (el cuadro 5 de fracciones simétricas justificará por qué no siempre optamos por la primera en orden lexicográfico de cada pareja).

Cuadro 3: Fracciones a considerar desde la casilla 28.

Salto 1	Último salto
11	13
11	22
11	34
11	38
11	43
11	45
13	18
13	22
13	34
13	38
13	43
13	45
18	34
18	43

Cuadro 3: (continúa)

Salto 1	**Último salto**
18	45
22	18
22	34
22	38
22	43
22	45
34	43
34	45
38	18
38	34
38	43
38	45
43	45

Pero, aunque el número de soluciones cerradas será la suma de las soluciones obtenidas para estas 27 fracciones, no hay que explorarlas todas. Algunas de estas fracciones son simétricas entre ellas y, si bien conducen a soluciones formalmente distintas, el número es el mismo para una fracción y su simétrica, por lo que basta con analizar una de ellas.

En primer lugar, hay seis parejas, que se muestran en el cuadro 4, donde las dos fracciones son simétricas respecto de la diagonal principal, la que va de la casilla 1 a la 64. De las 21 fracciones que quedan, aún hay seis parejas más, las que aparecen en el cuadro 5, en las que las dos fracciones son simétricas entre sí.

Cuadro 4: Pares (izquierda y derecha) de fracciones simétricas respecto de la diagonal principal partiendo de la casilla 28.

Salto 1	**Último salto**	**Salto 1**	**Último salto**
11	13	18	34
11	22	18	43
11	38	18	45
13	22	34	43
13	38	34	45
22	38	43	45

Cuadro 5: Otros pares (izquierda y derecha) de fracciones simétricas partiendo de la casilla 28.

Salto 1	**Último salto**	**Salto 1**	**Último salto**
11	34	13	18
11	43	22	18
11	45	38	18
13	43	22	34
13	45	38	34
22	45	38	43

En definitiva, partiendo de la casilla 28 sólo se necesita explorar las 15 fracciones que podemos ver en el cuadro 6, en lugar de las 24 necesarias cuando se parte de la casilla 1. Pero lo más importante es que, como veremos en la siguiente sección, el número de soluciones encontrado para cada fracción es similar en todas ellas. El tiempo empleado para explorarlas ha sido 60 días.

Cuadro 6: Únicas fracciones que es necesario explorar desde la casilla 28.

Salto 1	**Último salto**
11	13
11	22
11	34
11	38
11	43
11	45
13	22
13	34
13	38
13	43
13	45
22	38
22	43
22	45
38	45

3. Resultados

El número de soluciones cerradas encontradas es **13 267 364 410 532**. El cuadro 7 recoge las soluciones encontradas para las distintas fracciones que parten de la casilla 1. En la figura 10 se representa la misma información de manera gráfica.

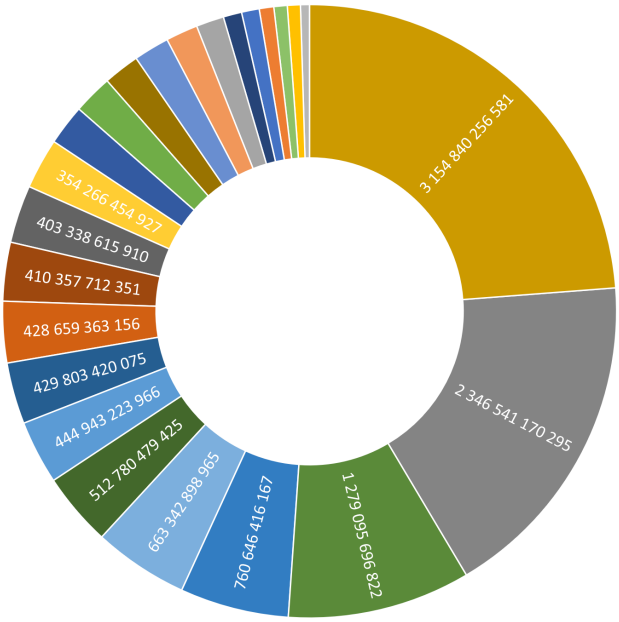

Figura 10: Número de soluciones según la fracción cuando todas comparten el primer salto de la casilla 1 a la 18.

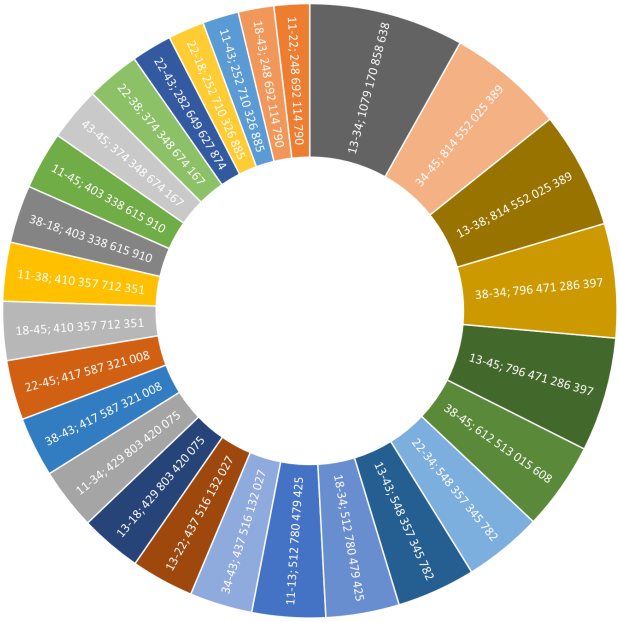

Figura 11: Número de soluciones para las fracciones relevantes que parten de la casilla 28.

Cuadro 7: Soluciones encontradas para las distintas fracciones que parten de la casilla 1.

Inicial	Salto 1	Salto 2	Salto 3	Soluciones
1	18	3	9	2 346 541 170 295
1	18	3	13	428 659 363 156
1	18	3	20	282 813 623 549
1	18	12	2	663 342 898 965
1	18	12	6	354 266 454 927
1	18	12	22	63 051 360 055
1	18	12	27	93 397 914 776
1	18	12	29	224 118 832 080
1	18	28	13	429 803 420 075
1	18	28	22	252 710 326 885
1	18	28	34	512 780 479 425
1	18	28	38	403 338 615 910
1	18	28	43	248 692 114 790
1	18	28	45	410 357 712 351
1	18	33	27	1 279 095 696 822
1	18	33	43	760 646 416 167
1	18	33	50	3 154 840 256 581
1	18	35	20	90 971 583 231
1	18	35	25	444 943 223 966
1	18	35	29	194 325 978 285
1	18	35	41	274 648 965 977
1	18	35	45	101 414 329 726
1	18	35	50	128 910 152 866
1	18	35	52	123 693 519 672
Total				**13 267 364 410 532**

Como puede verse, cuando se parte de la casilla 1, el número de soluciones es muy distinto de unas fracciones a otras; el número de soluciones de la fracción 1-18-33-50 es seis veces el de la media. Esto hace que la estrategia de dividir la búsqueda en fracciones para aprovechar el cálculo paralelo que permite un ordenador con múltiples procesadores pierda utilidad. La mayoría de las fracciones se completan mucho antes que la 1-18-33-50 que se convierte en la etapa crítica del proceso. Lo que inicialmente se preveía que iba a tardar 50 días, hubiera requerido 300 de no haberse subdividido, como se hizo, en más fracciones.

En la práctica, resultó más eficiente partir de la casilla 28. Los resultados, para las 27 fracciones del cuadro 3 —que son las que hay que sumar para encontrar el total de ciclos hamiltonianos—, aparecen en el cuadro 8 y la figura 11.

Cuadro 8: Número de soluciones para las fracciones relevantes que parten de la casilla 28. Se han explorado las 15 fracciones del cuadro 6 y se han completado las demás por simetría.

Salto 1	**Último salto**	**Soluciones**
11	13	512 780 479 425
11	22	248 692 114 790
11	34	429 803 420 075
11	38	410 357 712 351
11	43	252 710 326 885
11	45	403 338 615 910
13	18	429 803 420 075
13	22	437 516 132 027
13	34	1 079 170 858 638
13	38	814 552 025 389
13	43	548 357 345 782
13	45	796 471 286 397
18	22	252 710 326 885
18	34	512 780 479 425
18	38	403 338 615 910
18	43	248 692 114 790
18	45	410 357 712 351
22	34	548 357 345 782
22	38	374 348 674 167
22	43	282 649 627 874
22	45	417 587 321 008
34	38	796 471 286 397
34	43	437 516 132 027
34	45	814 552 025 389
38	43	417 587 321 008
38	45	612 513 015 608
43	45	374 348 674 167
Total		**13 267 364 410 532**

La aplicación obtiene los mismos resultados partiendo de dos casillas distintas, verificando así el buen funcionamiento de la misma.

Para finalizar, el cuadro 9 muestra el número de intentos que, por término medio, ha realizado la aplicación hasta obtener una solución, así como el número del salto en el que los intentos fallan porque, o bien no había casilla a la que saltar (atasco), o bien apareció una casilla en isla o una casilla colgante. Nótese que el número de saltos hasta un atasco no es el número medio de saltos de un recorrido: algunos recorridos habrían llegado más lejos de no haberse implementado la poda por isla o colgante. Los valores varían según se parta de una casilla u otra. Aquí se muestran para las casillas 1 y 28, que son las exploradas en este trabajo.

Cuadro 9: Estadística de intentos.

Casilla inicial	**Intentos por solución**	**Atasco**	**Isla**	**Colgante**
		Número de salto en que sucede		
1	43,3	51,0	47,8	46,6
28	52,1	49,4	46,4	45,6
Casilla inicial	**Intentos por solución**	**Atasco**	**Isla**	**Colgante**
		% Intentos cancelados/Intentos totales		
1	43,3	3,3 %	3,2 %	91,3 %
28	52,1	4,5 %	3,7 %	89,8 %

Estos valores deben considerarse como indicativos, ya que el número de intentos depende de factores como el orden en el que se consideran los movimientos desde cada una de las casillas, y hemos visto que puede conducir a resultados muy diferentes. Los valores corresponden a la exploración de los movimientos en el sentido de las agujas del reloj. Las columnas de la derecha muestran el porcentaje de intentos que se cancelan por las tres causas que contempla la aplicación (el resto, hasta 100, es el número de soluciones). El 90 % de los intentos se cancelan cuando la aplicación detecta una casilla colgante, lo cual explica la gran eficiencia de esta regla.

Referencias

[1] M. Löbbing y I. Wegener, The number of knight's tours equals 33, 439, 123, 484, 294 — counting with binary decision diagrams, *Electronic J. Combin.* **3** (1996), no. 1, Research Paper 5.

[2] B. D. McKay, *Knight's Tours of an* 8×8 *Chessboard*, Technical Report TR-CS-97-03, Department of Computer Science, Australian National University, febrero de 1997.

[3] H. J. R. Murray, *A History of Chess*, Oxford University Press, 1913.

[4] E. Sandifer, *Knight's Tour*, entrada en la columna *How Euler Did It* de MAA Online, `http://eulerarchive.maa.org/hedi/HEDI-2006-04.pdf`, abril de 2006.

[5] H. C. von Warnsdorf, *Des Rösselsprunges einfachste und allgeneinste Lösung*, Th. G. Fr. Varnhagensehen Buchhandlung, Schmalkalden, 1823.

Alberto Bañón Serrano, C/ Sebastián de la Plaza n.º 2, E-1, 5-F, 28805 Alcalá de Henares. Madrid
Correo electrónico: `alberto@interajedrez.com`